March 2017

An Exploration of Maximum Power Point Tracking Algorithms

Andrew F. Flynn
Worcester Polytechnic Institute

Benjamin James Beauregard
Worcester Polytechnic Institute

Johnathan Werber Adams
Worcester Polytechnic Institute

AN EXPLORATION OF MAXIMUM POWER POINT TRACKING ALGORITHMS

Johnathan Adams
Benjamin Beauregard
Andrew Flynn

Advisor:
Professor A. Emanuel

March 6, 2017

WPI

Abstract

This project investigated the performance of various solar maximum power point tracking algorithms on a single hardware platform. It also developed a low-cost hardware platform including a rudimentary solar cell emulator circuit and a maximum power point tracker circuit to test these algorithms. Additionally, the project covered the development of the two software algorithms that were tested in the project: the "Sweep" algorithm and the "Perturb and Observe" algorithm.

Acknowledgements

We would like to express our sincere gratitude to Professor Emanuel for the guidance, wisdom, and kindness he provided us as we brought this project to fruition.

In addition we would like to thank the following individuals for their contributions to this project. Without their help this report would not have been possible.

Professor S. J. Bitar

Mr. James O'Rourke

Mr. Bill Appleyard

Contents

List of Tables

List of Figures

1 Introduction

As fossil fuels increase in scarcity and human civilization increases its global demand for electrical power, so-called "alternative" energy sources increase in importance and prevalence. Alternative energy doesn't necessarily mean renewable energy, it simply means that the source of energy isn't a traditional fossil fuel. Renewable energy sources are more politically controversial than other forms of renewable energy, but are increasingly prevalent, especially in situations where more traditional energy sources are not viable.

Solar power is one increasingly popular source of renewable energy, due to its low cost, scalability, and availability. In large-scale solar energy harvesting operations, sunlight is used to boil water to drive steam turbines. More common, however, are photovoltaic panels that produce less power, take up less space, and require less management and upkeep to function [1].

Maximum power point tracking is of critical importance to photovoltaics because it brings an increase in supplied power without having to increase the area or weight of the photovoltaic array itself. As such, maximum power point tracking is part of most modern photovoltaic array controllers, though it is uncommon for companies to publish technical details relating to the maximum power point tracking hardware and algorithms.

1.1 Project Statement

This project's primary purpose was to investigate the efficiency of various maximum power point tracking algorithms on a single hardware platform. Ancillary to this purpose were: the development of a low-cost hardware platform with which to test these algorithms, the development of a low-cost solar cell emulator for the reproduction of solar conditions in a laboratory setting, and the implementation of the software algorithms to be tested. The project encompassed the design, simulation and implementation of both the hardware and software that made up the maximum power point tracker and the solar cell emulator, as well as the testing of two separate maximum power point tracking software algorithms for comparison against each other.

2 Background

2.1 Terminology

The following terms and acronyms are used throughout this paper:

Solar Power In the context of this project, solar power refers to electrical power derived from the use of one or more photovoltaic devices. There are other ways to harness power generated by the sun, but this project focuses only on photovoltaic devices.

Maximum Power Point (MPP) The maximum power point is a phenomenon made possible by the nonlinear nature of the photovoltaic current-voltage characteristic. It is a single voltage and current at which the photovoltaic outputs its maximum possible power for a given load. Figure 2.1 shows both the current-voltage characteristic and power-voltage characteristic of an ideal photovoltaic. Real photovoltaic characteristics are not as smooth, but retain the maximum power point and inverted logarithmic current-voltage characteristic.

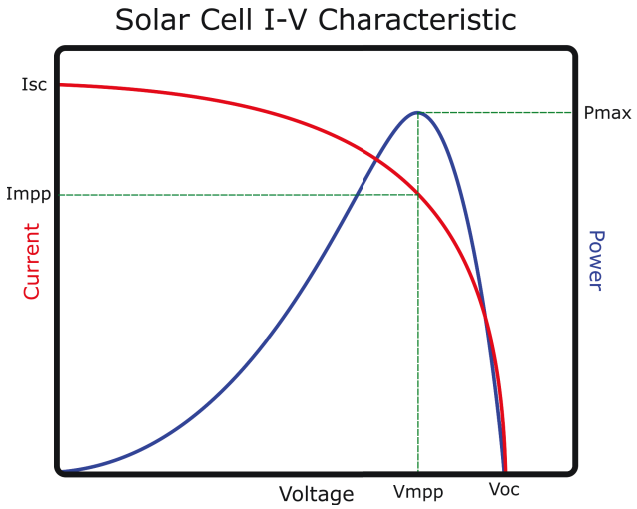


Figure 2.1: Example of a Photovoltaic Current-Voltage and Power-Voltage Characteristic

Maximum Power Point Tracking (MPPT) Maximum power point tracking is a term to describe the process of actively varying circuit behavior so as to keep the photovoltaic outputting the maximum possible for any set of given conditions e.g. load impedance, solar angle, intensity, atmospheric refraction, etc.

Efficiency (In the context of this project) The efficiency of any given algorithm is defined by two factors: first, the accuracy with which it tracks the maximum power point and second, the speed and accuracy with which it responds to changes in the maximum power point. The efficiency of hardware modules is the difference in power at the input and the output of the module.

Pulse Width Modulation (PWM) Pulse Width Modulation can be used as a way of encoding information, or as used in this project as a way of controlling the power delivered to an electrical load. This method varies the proportion of on-time to off-time of a high frequency pulse. This proportion is referred to as the "duty cycle."

Proportional-Integral-Derivative (PID) Proportional Integral Derivative control is a control loop feedback algorithm. The algorithm gets its name from the three ways it attempts to achieve the set point, a proportional gain, an integral gain and a derivative gain. Not all terms are used in every system, and the needs of each system must be evaluated to determine the appropriate configuration.

2.2 Conceptual Background

Solar panels are used to convert solar radiation into electrical energy to power many of the electronics in our everyday lives. These panels are made up of a number of individual photovoltaic cells in a large unit that produces a usable power output. Such panels are usually sold with a maximum power rating given by the maximum output current times the maximum output voltage, or $P = I * V$. Output voltage is generally determined by the number of photovoltaic cells placed in series with each other, and is thus generally limited by the overall size of the panel. Based on the equation for power above, the output power of a photovoltaic array is determined by its generated current. This can be achieved by utilizing different crystal structures in the panels themselves, or by improving the efficiency of the cells.

Photovoltaic devices behave fundamentally differently from both ideal power sources and common real power sources such as batteries or rotary generators. Whereas these sources have a linearly proportional relationship between output current and voltage, the relationship between a photovoltaic's output current and voltage is an inverted logarithmic curve. This behavior of the current vs. voltage characteristic is shown in Figure 2.1 on page 2. Also shown in Figure 2.1 is the power vs. voltage characteristic, illustrating how the maximum power point occurs at the "knee" of the current-voltage characteristic curve. "Most solar panel manufacturers will specify the panel voltage at maximum power (V_{mpp}). This voltage is typically around 70% – 80% of the panel's open circuit voltage (V_{oc})" [2].

In order to achieve repeatable test results over various times of day and atmospheric conditions, the testing of solar powered devices relies upon solar cell emulators. Commercial solar cell emulators are prohibitively expensive relative to this project's budget, which led to an investigation of emulator circuits that could be built for this project. Such models can vary in accuracy and complexity from a linear Current vs. Voltage characteristic [3] to the ability to model partially shaded solar panels [4].

The most basic design consists of a variable resistor in series with a DC power supply. Such a design does not produce the characteristic current vs voltage curve, but does produce a

distinct maximum power point [3]. A design that reproduces the "knee" in the current vs voltage curve of a solar panel utilizes a current source in parallel with a diode. This design can be made more accurate by including a resistance in parallel with the diode to represent any shunt resistance in the panel. A series resistance can also be included to model the equivalent series resistance of all of the components of a solar panel. This design improves upon the previous by attempting to model the non-linear current vs. voltage curve. The most complicated design utilizes a photodiode and an amplification circuit to accurately model the behavior of a solar cell. Multiple photodiodes can be configured to match the overall layout of a specific solar panel. By adding bypass diodes to the photodiodes, the partial shading behavior of a solar panel will also be emulated [4].

Existing work in the field of Maximum Power Point Tracking was evaluated and a number of algorithms were investigated. Algorithms evaluated include, Current Sweep, Perturb and Observe or Hill Climb, Incremental Conductance, Ripple Correlation Control, and Beta Method.

The **Current Sweep** algorithm varies the current from an open circuit to a short circuit in order to obtain the I-V characteristic of the panel under the current conditions. This information is then evaluated to find the maximum power point. The process is generally repeated at a fixed interval to maintain the maximum power point [5].

Perturb & Observe or **Hill Climb** is frequently the focus of academic papers [5]. This simple algorithm slightly adjusts the duty cycle of a power converter and uses the resulting change in output power to maximize power, or "climb the hill" defined by the Power vs. Voltage characteristic of a solar panel.

Incremental Conductance uses the derivative of the power vs the derivative of the voltage to determine how to adjust the duty cycle of a DC-DC converter. The algorithm can be simplified to compare the incremental conductance ($^{\Delta I}/_{\Delta V}$) to ($^{I}/_{V}$) for decision making [5]

Ripple Correlation Control attempts to correlate the phase of the power and voltage from a solar panel found in the small ripple introduced by a DC-DC switching converter. The phase information can be found by finding the product of the derivative of power and voltage [6]. By continuously integrating this value it can be used to quickly adjust the duty cycle to achieve the maximum power point.

The **Beta** algorithm relies upon a variable B that will monotonically increase with duty cycle of a DC-DC converter [7]. The calculated value is then used to find a new duty cycle that is closer to the maximum power point.

2.3 Prior Work

This section covers other projects and products that influenced this project. This includes major qualifying projects from previous years that explore solar energy or maximum power point tracking and commercial products that perform those operations.

Renewable Energy Applications (2011) [8]

The Renewable Energy Applications MQP focused on the design and development of a renewable energy system at WPI's Atwater Kent Laboratories. Two solar panels were installed on the roof of Atwater Kent and power from these panels was collected with a custom MPPT circuit, which employed the perturb and observe algorithm to charge a lead-acid battery. The Renewable Energy Applications MQP concentrated primarily on operational theory, and potential configurations for a renewable energy monitoring system that could be permanently installed and benefit the WPI community. Emphasis was also placed on ensuring the flexibility of the MPPT circuit. The broad scope of the Renewable Energy Applications MQP influenced this project to focus specifically on solar maximum power point tracking.

Grid-Independent Charging Station with Power Flow Display (2012) [9]

The Grid-Independent Charging Station MQP designed and constructed a functioning prototype solar charging station which provided both 5V DC (via USB), and 120V AC (via a NEMA 5-15R 15A duplex receptacle). The solar power was provided by the panels installed on the roof of Atwater Kent Laboratories from the Renewable Energy Applications MQP. The Grid-Independent Charging Station MQP used an off-the-shelf MPPT battery charging module due the the abandoned state of the Renewable Energy Applications project. Combining other off-the-shelf products with custom arduino-based hardware, the Grid-independent Charging Station MQP constructed a system capable of displaying the direction and magnitude of power flow in the charging system. The Grid-Independent Charging Station MQP's discussion of the Renewable Energy Applications MQP influenced this project to design and construct a custom mppt circuit from scratch rather than attempt to build upon a past MPPT related MQP.

Solar Charging Station (2014) [10]

The Solar Charging Station MQP sought to develop a prototype solar charging station based on a patio umbrella with built-in solar panels. The Solar Charging Station MQP set itself apart from products on the market by incorporating maximum power point tracking in the design of the flyback converter-based charging circuit, which charges a battery to power a 5 V DC and 120 V AC output. The perturb and observe algorithm was selected for use in the Solar Charging Station project. The Solar Charging Station MQP's goal of developing a consumer product device influenced this project to focus on a device for research rather than for a comercial market.

Maximum Power Point Tracker (2014) [11]

The Maximum Power Point Tracker MQP designed and constructed a custom MPPT specifically for an off-grid charging application, and implemented the final product in the Grid-Independent Charging Station from 2012. Similar to the three previously discussed MQPs, the Maximum Power Point Tracker MQP placed emphasis on idea conception, operational theory, and system integration. This project builds on the theoretical work of the Maximum Power Point Tracker MQP as well as the three other previously mentioned MQPs in order to provide a solid practical implementation of a maximum power point tracker.

2.4 Market Research

The group researched the availability and functionality of Maximum Power Point Trackers on the market today. These devices are often used to charge 12V and 24V battery systems and incorporate specialized charging features to improve the value of the device. Not all of the devices claiming to use MPP Tracking provide information on what algorithm is used for maximizing power output. It is possible that this information wasn't deemed important, or considered a trade secret. The lack of information on the underlying algorithm makes it difficult to assess whether these products could be improved, but it is likely that a more advanced algorithm could prove beneficial if implemented in such a solar powered battery charge controller.

3 Hardware Design

This section covers the design, simulation, prototyping, and testing of the hardware for the Maximum Power Point Tracker project. The hardware for this project was meant as a test bench for various algorithms, rather than as a practical, in-place solution to charge a battery from an actual solar cell. As such, the design constraints are slightly more relaxed than those of a MPPT for use in a production system. Table 3.1 was made by cross-referencing a list of available components from Digi-Key and specifications from the results of the group's market research.

Table 3.1: Absolute Maximum Ratings

Absolute Maximum Ratings			
Symbol	Parameter	Value	Unit
Vin	Input Voltage to MPPT	30	V
Vout	Output Voltage from MPPT	30	V
Iin	Input Current to MPPT	1.6	A

In addition to these absolute maximum ratings, the system had a number of qualitative design guidelines. The first of these was the ability to power itself from its own output. Secondly, the system would be controlled digitally by some sort of microcontroller. Another goal was the ability to program the MCU in situ. Also, programmable over-voltage protection for the output side of the MPPT circuit was desired. The MPPT circuit was also to be able to handle small or medium solar panels, should the opportunity to test it with one arise.

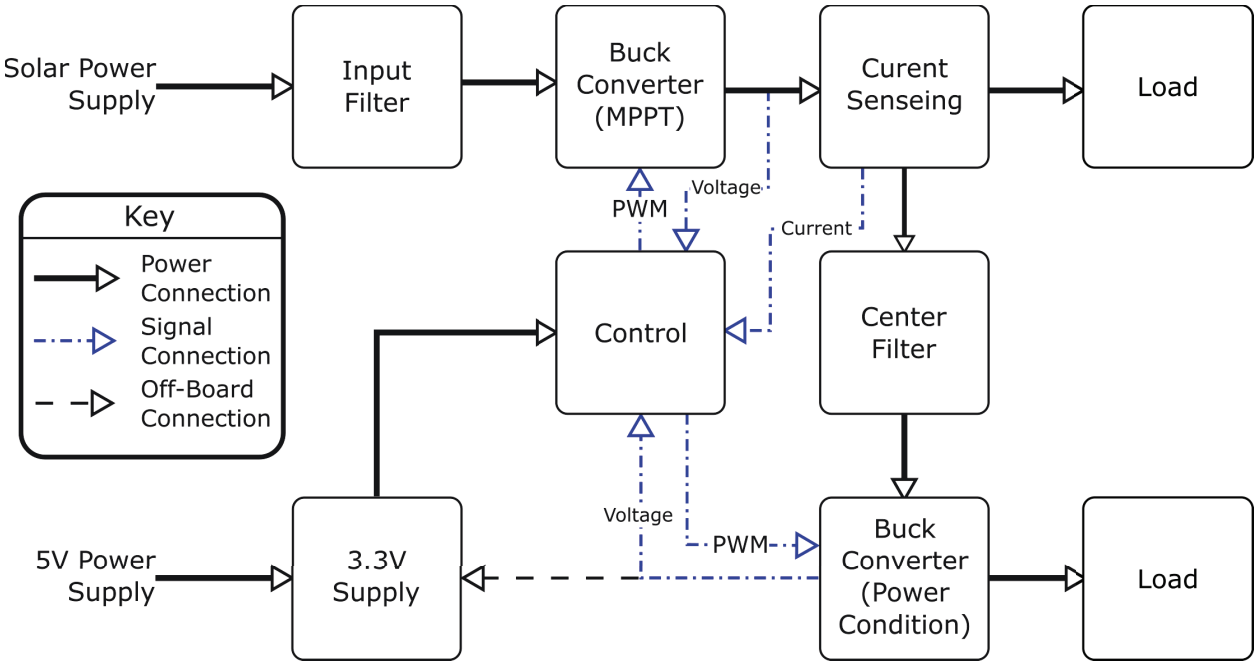


Figure 3.1: Hardware Block Diagram

Overall, the design of the circuitry follows the block diagram shown in Figure 3.1 on page 7. As in the figure, the buck converter and filter blocks are used multiple times for different purposes. The second buck converter was added as part of a plan to make the MPPT power itself. This plan was scrapped due to lack of time, though the hardware was left in place so that it could be used in the future for expansions on this project if desired.

3.1 Theory

This subsection covers the theory of operation behind each part of the maximum power point tracker as well as some of the mathematics applied in their design.

Solar Cell Emulator

The Solar Cell Emulator circuit is not part of the Maximum Power Point Tracker. It is an external test fixture that happens to be central to this project. Winter conditions in New England are not reliably sunny enough for proper rigorous testing of the MPPT with an actual solar panel. There are purpose-built solar panel emulators that can simulate all sorts of behaviors of solar panels, but all of these proved prohibitively expensive for the budget this project was allotted. Thus, in order to test the MPPT circuit, a custom-built, less powerful, and more cost-efficient test fixture was needed. The Solar Cell Emulator does not seek to fully emulate the behavior of a solar cell, rather it simply seeks to reproduce the nonlinear characteristic current-voltage relationship that solar cells are known for. Crucially, this creates a Maximum Power Point for the MPPT circuit to track, shown in Figure 3.2.

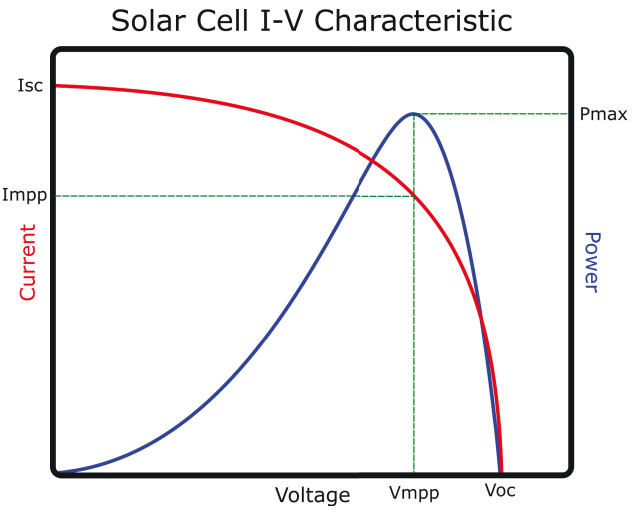


Figure 3.2: Example of a Photovoltaic Current-Voltage and Power-Voltage Characteristic

The original solar cell emulator circuit designed by Professor Bitar is shown in Figure 3.3 on the next page. It uses a MOSFET as a voltage-controlled resistor to create a current-voltage characteristic curve with the same inverted logarithmic shape as one from a solar panel.

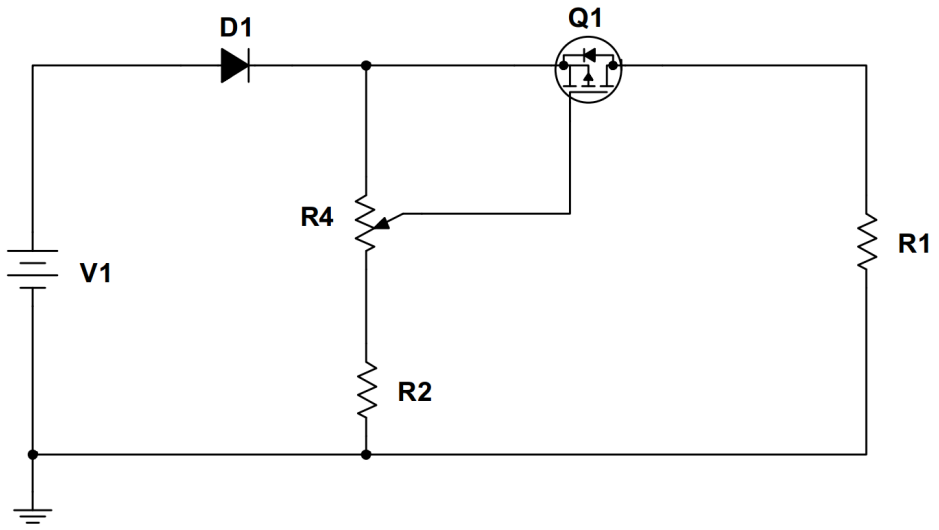


Figure 3.3: Original Solar Cell Simulator Schematic

The diode, D1, is a reverse current protection device and protects the emulator circuit from laboratory errors setting up the power supply. The gate voltage of the P-channel MOSFET, Q1, is determined by the potentiometer R4 and the resistor R2 in the following equation:

$$V_{SG} = \left(1 - \frac{R2 + (1-P)R4}{R2 + R4}\right) V_{D1} \tag{3.1}$$

Where P represents the decimal percentage of the R4 Potentiometer wiper. The resistor R1 represents the load, which in this project is the MPPT circuit. The voltage source, V1 represents the laboratory power supply.

Buck Converter

The Maximum Power Point Tracker accomplishes changes to its current and voltage throughput by varying the duty cycle of a buck converter's switch. Other MPPTs use more complex DC-DC converters that frequently result in increased efficiency at the cost of increased design complexity. As none of the team members had any prior experience with DC converters, the simple design and theory associated with buck converters was preferable to more complex converters. This reduced the chances of fundamental design errors and made debugging the circuit a more intuitive process. The converter was designed to operate at 50kHz which required larger and more expensive components than would be necessary at higher frequencies, but resulted in lower switching losses. This simplified the selection of a transistor, as the primary concern became on-resistance and the various parameters that relate to switching losses became secondary concerns.

This particular implementation of the buck converter circuit includes a high-side driver integrated circuit (U1). This takes an input from the control circuit (Figure 3.7 on page 13). This input is a square wave between 0V and 3.3V. The high-side driver biases this signal so

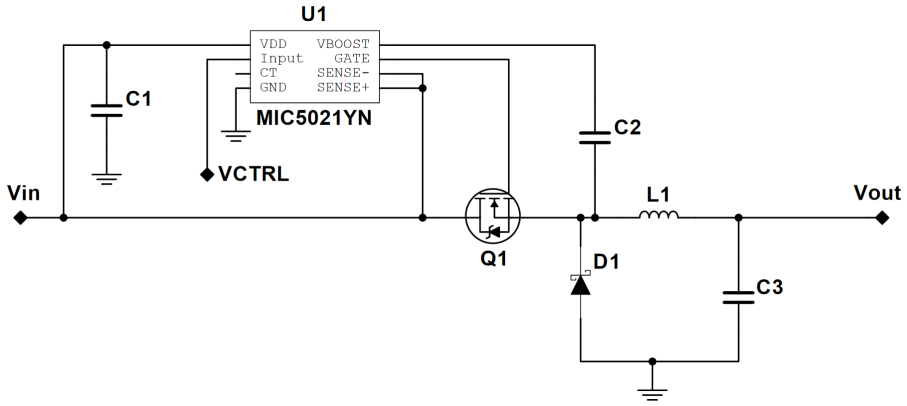


Figure 3.4: Original Buck Converter Circuit Schematic

that it lies between Vin and Vin + 3.3V. This allows the VCTRL signal to control the N-Channel MOSFET switch (Q1) despite the its source pin's connection to a net with a higher potential level than the control signal reaches. C1 is a filter capacitor for the power supply to U1. C2 is a "Boost Capacitor," connected as specified by the MIC5021YN datasheet (Appendix H.1 on page A-40). L1 serves to stabilize Vout by flattening out and averaging the switching from Q1, and D1 is a flyback diode so that current can flow through the load unimpeded even when the switch is open. L1 is chosen based on the following:

$$L_{min} = \frac{(1 - D)R_{Load}}{2f} \tag{3.2}$$

Equation (3.2) specifies the minimum possible inductance for the circuit, so the result of the equation should be rounded up to the nearest available component. More information about buck converter design can be found in Power Electronics by Hart [12]. D1 has less concrete requirements, though it has to be able to handle the maximum load current and should switch fast enough to keep up with the switch. For this reason, D1 was chosen as a 15SQ100TR Schottky diode. C3 is the output capacitance, a nonpolarized polymer capacitor with a "fast" transient response. Traditionally, these converters use a ceramic capacitor and an electrolytic capacitor instead of a single polymer capacitor, but using the polymer capacitor allows a larger bulk capacitance than ceramic capacitors without sacrificing transient response, to the point that theoretical models showed that with just the polymer capacitor and no bulk aluminum capacitor the output voltage ripple would still be less than 500mV peak to peak:

$$C_{min} = \frac{(1 - D)V_{out}}{8LV_{ripple}f^2} \tag{3.3}$$

Equation (3.3) calculates the minimum possible capacitance to achieve the desired output ripple voltage in an ideal converter. Actual component values should be above this number, rather than simply equal to it. Both Equation (3.2) and Equation (3.3) are dependant on duty cycle, D, and switching frequency, f. C_{min} must be calculated after an inductance is chosen because it depends on the inductor selection.

A second identical buck converter is used to reduce the MPPT voltage to within the input specification of the linear regulator. This was part of an "if there is time" feature that was ultimately not fully implemented where the MPPT would be able to power itself directly from its solar panel. More details about this are available in Section 3.3 on page 25.

Filter Network

The filter network was devised to protect delicate solar cells from the 50kHz switching transients generated by the buck converter. Solar cells function best when under constant load as opposed to an intermittent one [13]. The filter is used again between the MPPT's output and the second buck converter so that the MPPT output voltage stays at a relatively constant DC level for easy sampling by the microcontroller. This second filter is identical to the first filter because its purpose is identical: to reduce the effect of switching noise on the power line at a sensitive point, though in this case it is for better measurement rather than protection of the solar panel.

Taking into account that both of the filters lie on power supply lines, traditional techniques such as simple R-C, R-L and R-L-C filters proved inadvisable due to the necessity of a large resistance in series with the power line or a small one in parallel. Additionally, simple active filter designs proved problematic due to the traditional power-disconnect between input and output. To maintain design simplicity and achieve acceptable filtering with minimal series resistance on the power line, the team chose to design a multi-stage L-C filter, seen in Figure 3.5. The original design generated by NI Mutisim's filter wizard called for a five-stage L-C filter. The filter wizard makes no attempt to match real-world component values, so this filter was considered a starting point, and subsequently tweaked through simulation to produce a final design.

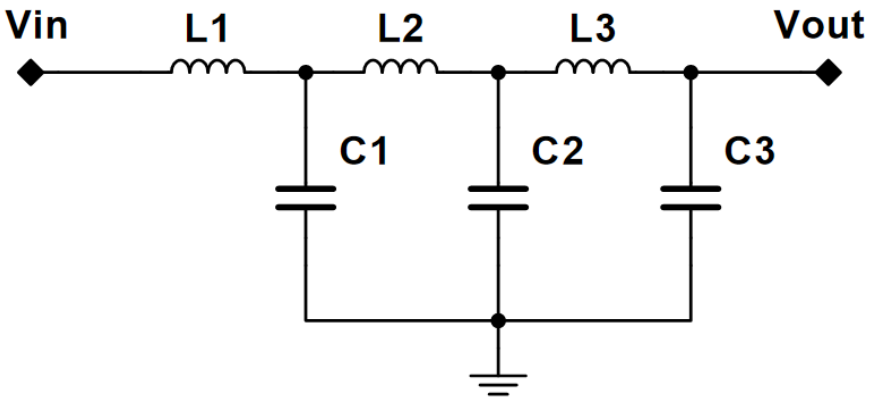


Figure 3.5: Filter Circuit Schematic

Current Sense Circuit

The current sensing circuit started out as a single-chip interface between the microcontroller (Figure 3.7 on the following page) and the output of the MPPT buck converter (Figure 3.4 on page 10). It grew as its input and output specifications changed into its present form. The circuit is based on a Hall-Effect current sensor integrated circuit. When initially making specifications for the circuit a more traditional current-sense resistor was considered, but the Hall-Effect current sensor was chosen because it was theoretically price-competitive and theoretically simpler to implement. The actual cost ended up higher then estimated, as the originally chosen current-sense chip was unavailable for purchase. The substitute chip had a larger current handling capability and as such its output used less than half of the microcontroller's ADC's dynamic range. To fix this problem the team inserted a non-inverting operational amplifier between the current sensor and the microcontroller. The resulting circuit is shown in Figure 3.6.

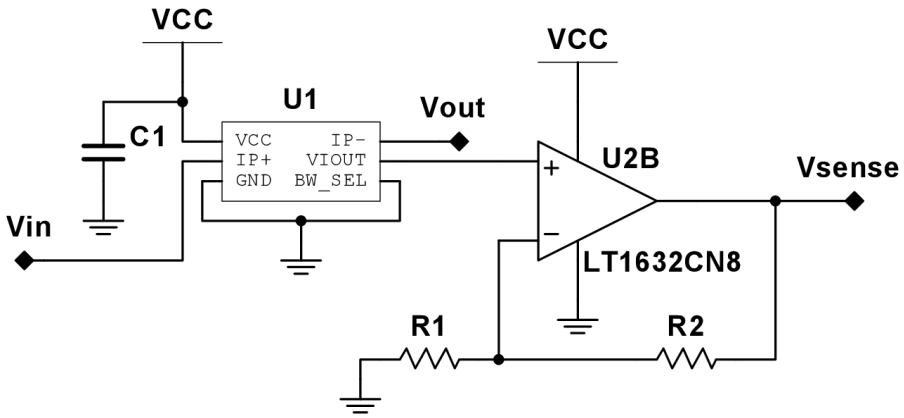


Figure 3.6: Original Current Sense Circuit Schematic

The operational amplifier had a specified gain of 3 to reconcile the current sensor's range with the microcontroller's dynamic range. This is determined by the feedback resistors, R1 and R2:

$$A_V = 1 + \frac{R2}{R1} \tag{3.4}$$

Control Circuitry

The control circuitry in this case is interface circuitry for a MicroController Unit (MCU). This interface circuitry primarily takes signals too large for the microcontroller's range and scales them to fit within that range. Additionally, it provides power conditioning for the MCU and human interface devices.

The first step of the design process for this circuit was selection of a MCU, as that sets dynamic range requirements and operating conditions. The team chose a variant of Texas Instruments' MSP430 Ultra-Low-Power Mixed-Signal Microcontroller (U1) because all of the

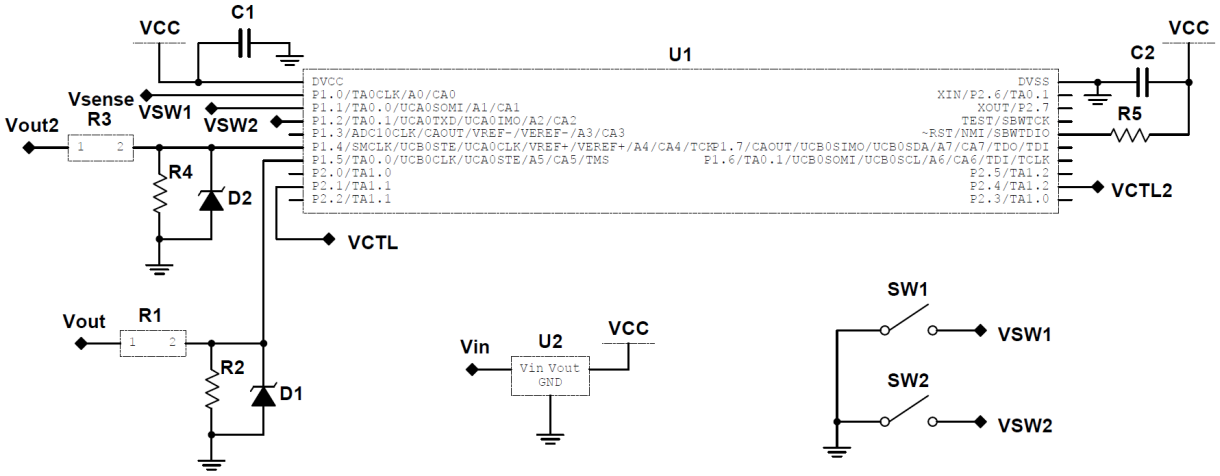


Figure 3.7: Original Control Circuit Schematic

team members had prior experience with the platform and knew it provided the versatility required, as well as a familiar programming interface. The project required two 16 bit timers, with at least one capable of controlling two GPIO pins, an Analog to Digital Converter (ADC) with at least 10 bits of quantization and availability in a DIP package to ease prototyping. The team decided on the MSP430G2553IN because it fulfilled all of these requirements.

The second step of the design process was to provide a power supply for the microcontroller, which required a low-current 3.3V supply. This was accomplished with a linear regulator connected to an external 5V supply. This allowed the addition of an external circuit that could control whether the MCU was powered from the onboard DC converters or an offboard power supply depending on power availability from the "solar panel," though there was not time to implement this functionality. The linear regulator chosen was the L78L33 (U2) from ST Microelectronics. Section 3.3 on page 25 covers changes made to this circuit after testing.

The recommended operating configuration of the microcontroller includes a filter capacitor on the 3.3V supply rail. This circuit uses a 100pF ceramic capacitor (C1) for that purpose. Additionally, in order for the MCU to operate it needs an enable signal. This is accomplished by tying a specific pin high via R5 in this circuit. C2 here is identical to and serves the same purpose as C1 does for the power supply to the MCU.

The circuit includes two identical voltage dividers, for the input signals Vout (the output voltage of the MPPT) and Vout2 (the output voltage of the second buck converter) respectively. These voltage dividers encompass resistors R1, R2, R3 and R4. These voltage dividers ensure that the expected values of these signals lie within the dynamic range of the MCU's ADC. The zener diodes, D1 and D2, serve to protect the MCU in the case that one or both of the input signals exceeds its expected value, driving the output of the voltage divider

above the absolute maximum voltage rating of the microcontroller, shown in Equation (3.5).

$$V_{out} = \begin{cases} \frac{R2}{R1+R2}V_{in} & \frac{R2}{R1+R2}V_{in} \leq 3.3\text{V} \\ 3.3\text{V} & \frac{R2}{R1+R2}V_{in} > 3.3\text{V} \end{cases} \tag{3.5}$$

The values of the resistors were chosen as R1 = R3 = 75kΩ and R2 = R4 = 750kΩ to give approximately a factor of 10 reduction in input signal magnitude. The microcontroller uses this information, combined with a voltage signal representing the current flow through the MPPT (Vsense) to control the two buck converters. The signal VCTL is a PWM signal that controls the MPPT buck converter, while VCTL2 does the same for the output buck converter. The control block has two buttons attached for testing purposes, here shown as SW1 and SW2 respectively.

3.2 Simulation

Simulation of the hardware design was performed using Cadence PSpice. Each of the blocks of the block diagram in Figure 3.1 on page 7, other than the Control and 3.3V Supply blocks, was written out as a subcircuit. This allowed for easy reconfiguration of the overall circuit to explore its performance under various conditions. The simulation code, followed by the custom model library is available in Appendix D on page A-11. A schematic of the simulation subcircuits available in Appendix E on page A-18. The purpose of the simulation was to verify the performance of each block individually, and then together. Algorithms were not implemented in the hardware simulation due to the complexity of performing the required mathematical and logical operations in PSpice. For passive components, such as inductors and capacitors, ideal models were used, coupled with the non-ideal characteristics of the physical components selected; for example, a 1mH inductor is modeled as a 1mH inductance in series with the physical component's equivalent series resistance (ESR) of 137mΩ. Active components such as transistors and diodes were modeled using manufacturer-published device models.

Solar Cell Emulator

The solar cell emulator circuit was simulated using a 23V ideal source, a manufacturer-provided model for the IRF9520 power MOSFET and a manufacturer-provided model for the MBR1060 diode. The potentiometer was modeled as a three-terminal subcircuit consisting of two resistors in series. The value and ratio of the resistors can be set by parameters passed into the subcircuit. Several simulations were performed with the potentiometer ratio set to different values. These simulations consisted of a parameter sweep, where a load resistance was varied from 1μΩ to 100Ω in 100mΩ increments. The voltage across, and current through the load resistance were measured for each simulation. These values were then plotted on a graph (Figure 3.8 on the next page) to show how the circuit successfully emulates the characteristic voltage-current curves of a solar panel.

Pulse Width Modulation Generation for Simulation

A feedback system was not implemented in the buck converters, as the maximum power point tracking and duty cycle adjustments were to be performed in software rather than hardware. In simulation, either a static pulse width modulation (PWM) duty cycle or a stepped duty cycle was used. The static duty cycle was implemented using a pulse voltage source. The stepped duty cycle was considerably more challenging to implement. The goal behind the stepped duty cycle was to more accurately simulate how the circuit will operate when controlled by the MCU, which increments or decrements the duty cycle by 0.3125% rather than "sliding" linearly between duty cycles. In simulation, convergence issues often occur when sudden changes are made to voltages and currents, wherein two components connected together "disagree" on the value of the voltage or current at their connected point.

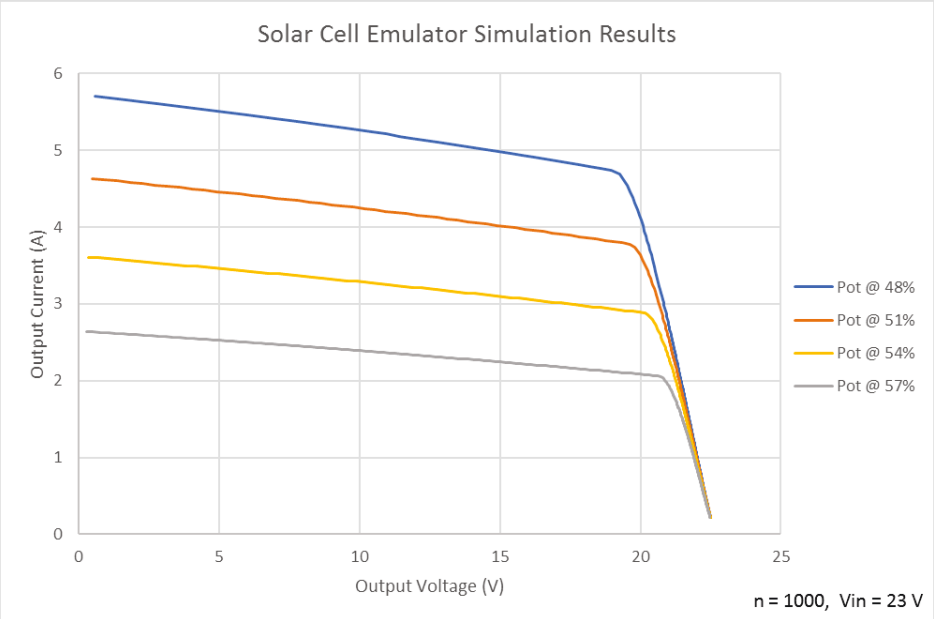


Figure 3.8: Solar Cell Emulator simulation results demonstrating voltage-current characteristics similar to those of a solar panel

Initially, a comparator circuit was used, which varied the width of a square wave based on the difference between a control voltage and a 50kHz triangle wave. The control voltage was generated using a piecewise-linear (PWL) voltage source. This method caused convergence issues which could not be resolved despite following numerous troubleshooting steps. An alternative solution was devised and implemented, using a fully defined PWM waveform, meaning that for each period of the 50kHz square wave, four voltage points were defined in the time domain. A python script was written to automatically generate the very lengthy PWL voltage source definition. The code for the python script is available at the end of Appendix D. The resulting voltage source was used to drive the gate of the buck converter MOSFET.

Buck Converter

The simulation of the buck converters did not stray far from the hardware design. Ideal passive components were used in conjunction with their respective ESR values. As discussed in the previous subsection on PWM generation, a feedback system for controlling the buck converters was not implemented in simulation, and the buck converter MOSFETs were instead controlled using a PWM signal. Several graphs were generated to demonstrate the performance of the buck converter in simulation. Figure 3.9 on the following page shows the PWM source successfully stepping through a range of duty cycles from 1% to 99% over 1.58sec. Below that, 3.10 shows a zoomed in graph of the buck converter output, allowing the transient response during a transition of duty cycle to be seen.

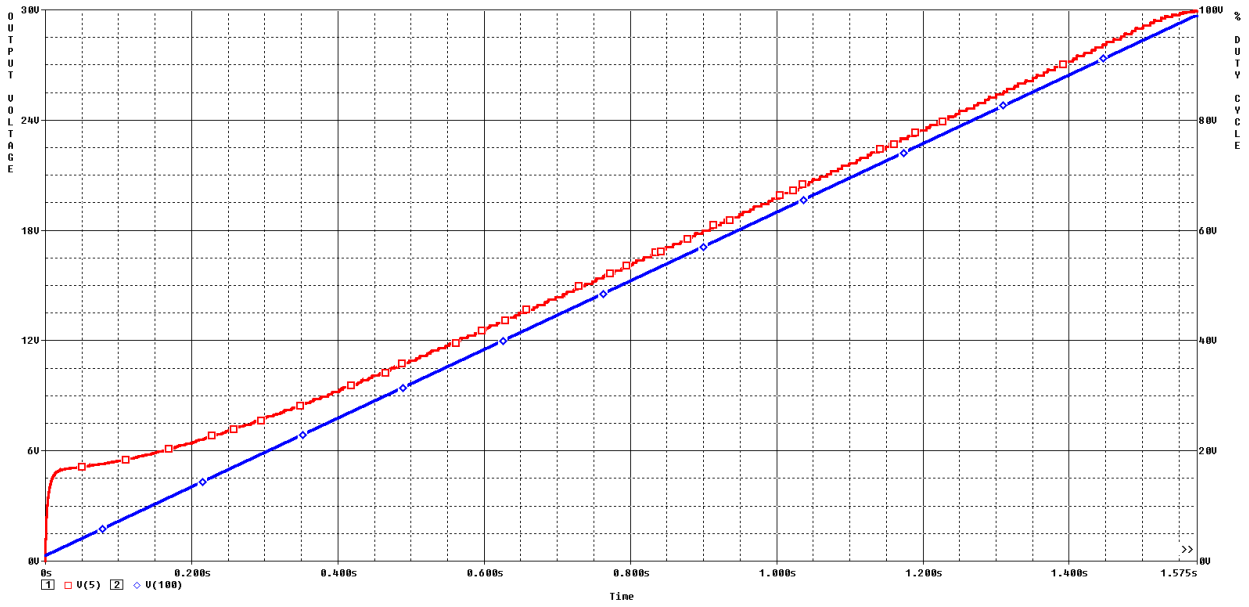


Figure 3.9: Buck converter simulation with stepped duty cycle showing output voltage (red, left axis), and duty cycle percentage (blue, right axis). Simulated using 30V source, 100Ω load, no filter

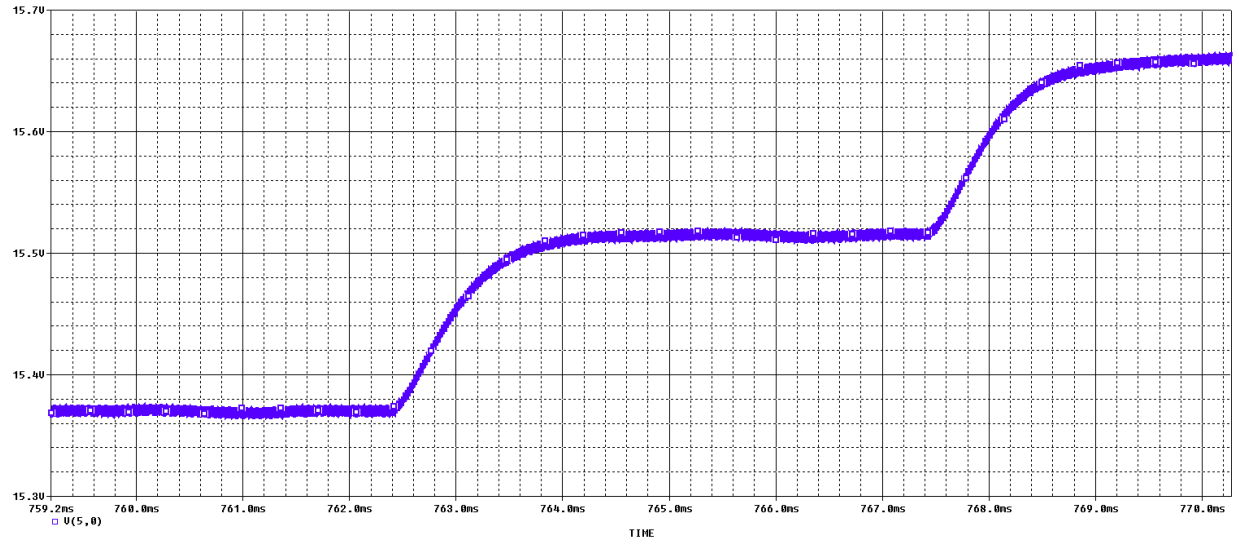


Figure 3.10: Buck converter simulation with stepped duty cycle showing a zoomed in view of the output voltage transient response during a duty cycle transition around 50%. Simulated using 30V source, 100Ω load, no filter

A hypothetical-efficiency analysis was performed using one buck converter connected directly to an ideal 30V source, and a 100Ω load resistance. Figure 3.11 shows the resulting efficiency $\left(\frac{P_{OUT\_AVERAGE}}{P_{IN\_AVERAGE}}\right)$ plotted over the duty cycle percentage.

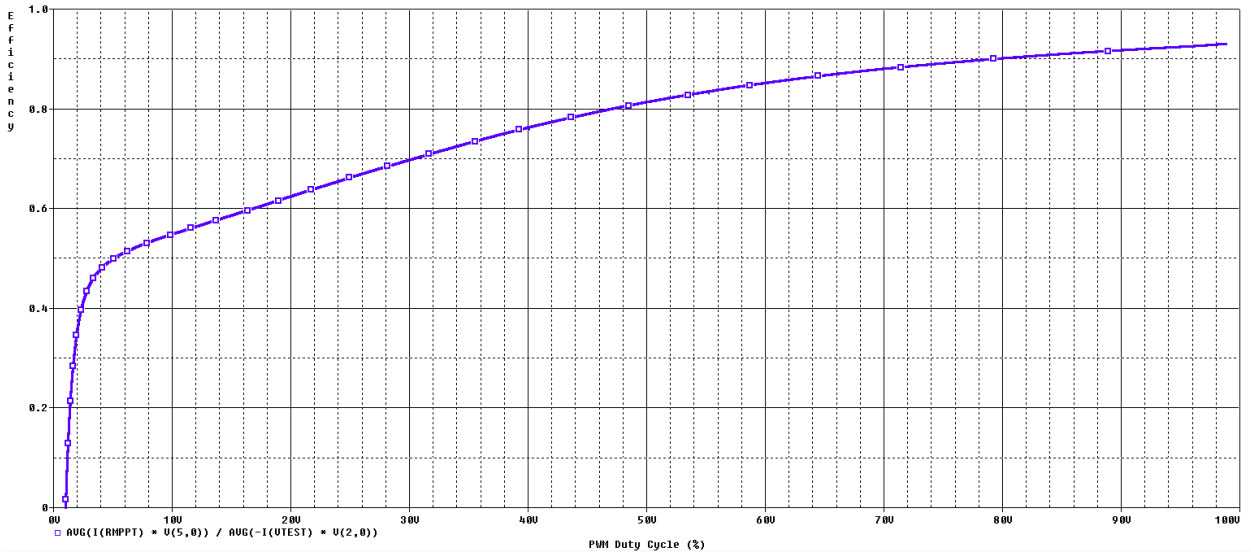


Figure 3.11: Buck converter simulation with stepped duty cycle showing Efficiency (P_{out}/P_{in} plotted over the duty cycle percentage. Simulated using 30V source, 100Ω load, no filter

Filter Network

Simulation of the filters was an integral part of the filter design process. The implementation of the filters in simulation uses the calculated inductances and capacitances combined with their respective equivalent series resistances. The first two stages of the filter are identical, and thus implemented using a subcircuit within the overall filter subcircuit. The filter performance was evaluated in the frequency domain, as seen in the Bode plot in Figure 3.12 on the following page. Signals in the 10kHz to 40kHz range have severely distorted phases, but that was not seen as a problem since: first, significant signals of those frequencies should not be present intentionally or otherwise in the circuit and second, the circuit does not need phase alignment of any sort to maintain its functionality. The filter performance was further evaluated in simulation in the time domain by placing a filter in series between a single buck converter and an ideal DC voltage source. A graph was generated, looking at the voltages after each stage of the filter to show the effect of each stage on the ripple. This graph is presented in Figure 3.13 on the next page, and is followed by Table 3.2 which explains the details of each trace in the graph.

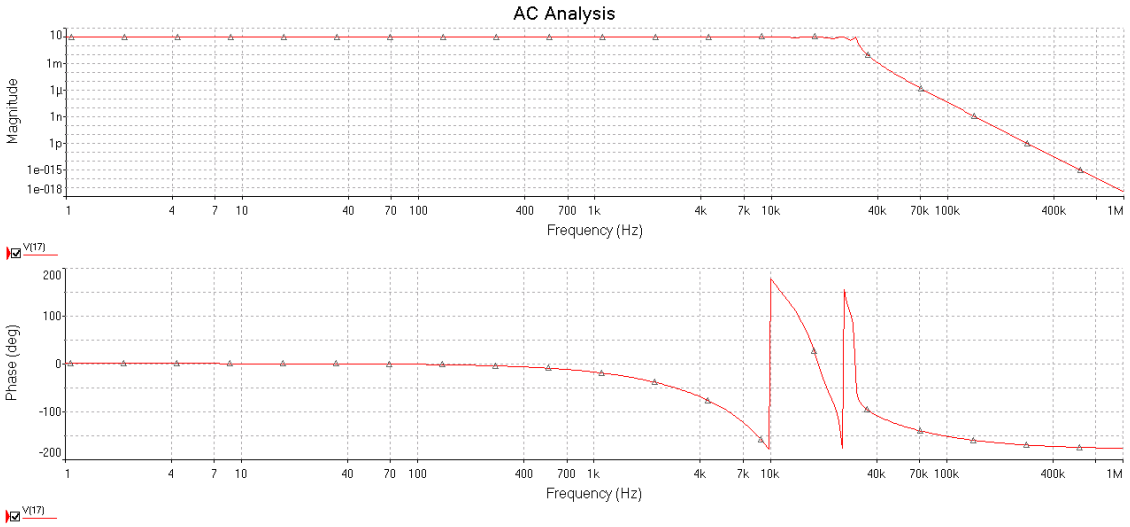


Figure 3.12: Bode plot for filter network

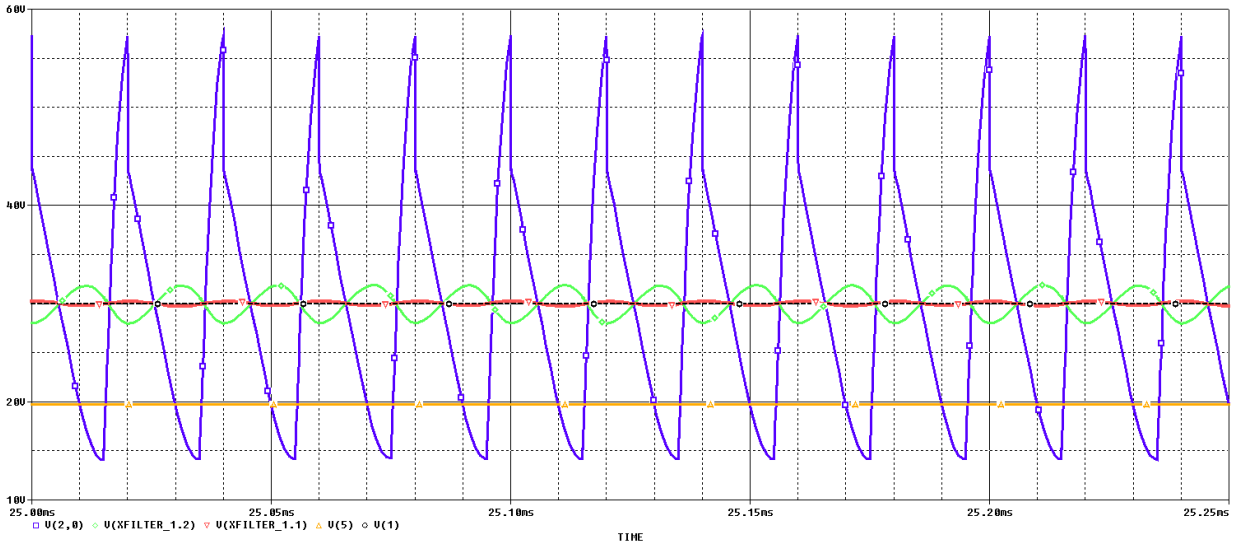


Figure 3.13: Graph of filter performance in simulation, shown by plotting voltages at each stage of the filter.

Table 3.2: Overview of traces in filter performance graph (Figure 3.13)

Trace Location	Trace Name	Color	ΔV (V)
Buck Converter Input	V(2)	Blue	43
Second filter stage output	V(XFILTER_1.2)	Green	3.82
First filter stage output	V(XFILTER_1.1)	Red	0.495
30v DC source	V(1)	Black	(N/A)
Buck Converter Output	V(5)	Yellow	0.0067

3.3 Implementation

This section covers the final implementation of the circuits described in the Theory and Simulation sections. It also covers changes made to circuits as a result of prototyping and testing.

Solar Cell Emulator

The solar cell emulator circuit went through some minor changes in order to allow for more accurate testing of circuits' responses to changes in solar conditions (Figure 3.14).

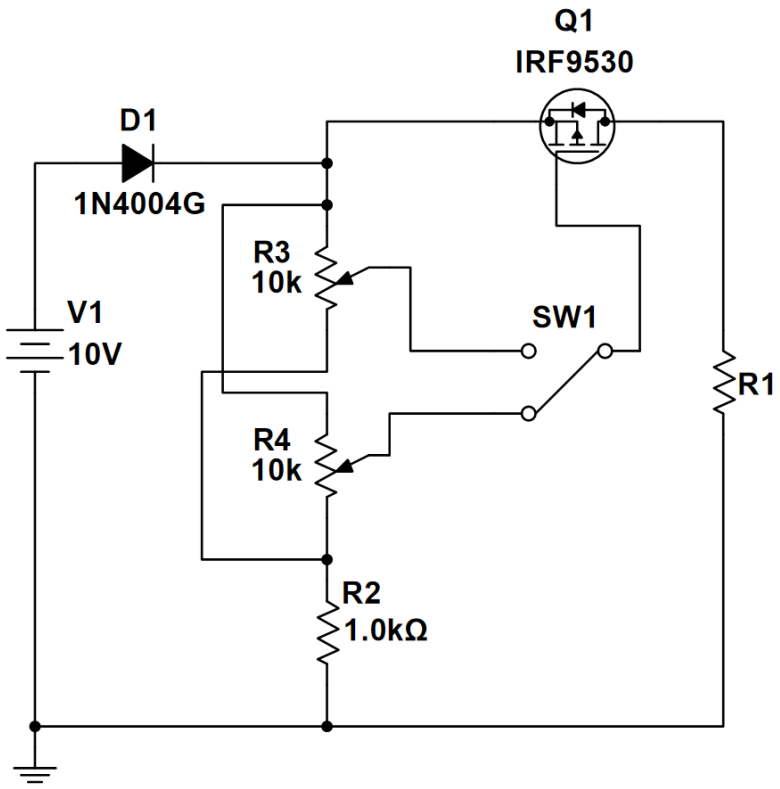


Figure 3.14: Final Solar Cell Emulator Circuit Schematic

The addition of the extra potentiometer and switch allows for two repeatable solar condition set-points so as to compare different algorithms' responses to the exact same change in solar conditions. As in Figure 3.3 on page 9, V1 repersents a standard adjustable DC power supply and R1 represents the load of the circuit.

Buck Converter

The buck converter circuit performed well in theory, but its output ripple voltage was higher than expected in practice. Figure 3.15 is an oscilloscope screen capture demonstrating this.

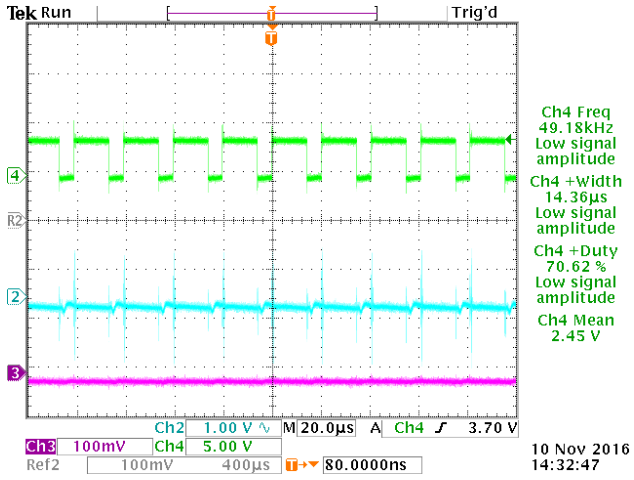


Figure 3.15: Output Voltage Ripple of Base Buck Converter Circuit [Blue]

This was solved by adding a 100µF aluminum electrolytic capacitor in parallel with the existing polymer output capacitor, as seen in Figure 3.16. The Zener diode is explained in Section 3.3 on page 23.

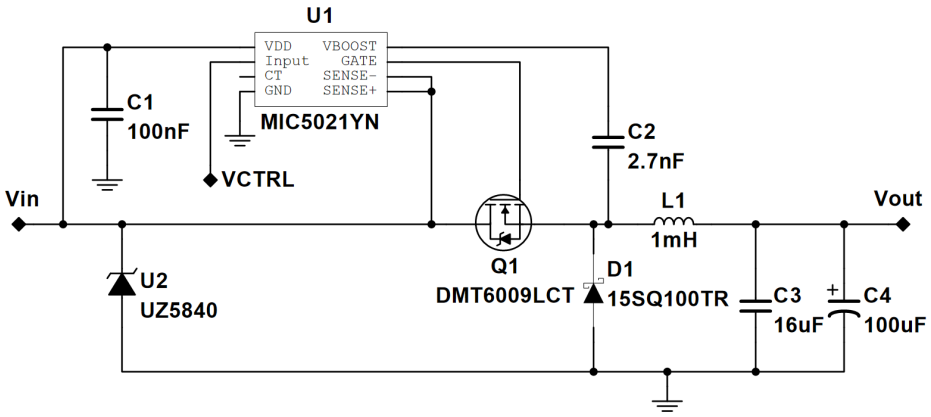


Figure 3.16: Final Buck Converter Circuit Diagram

Further testing revealed that the output stayed within its 500mV required dynamic range with the addition of the new 100µF aluminum electrolytic capacitor, C4. With this final parts configuration, the buck converter performed as shown in Figure 3.17 on the next page

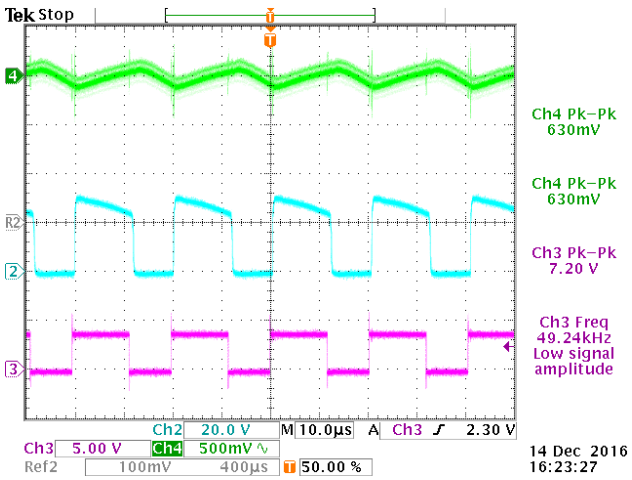


Figure 3.17: Output Voltage Ripple of Buck Converter Circuit with 100µF Capacitor [Green]

Filter Network

The use of the flyback diode (D1) and damping resistor (R1) is necessitated by the large total inductance (L1, L2, and L3 are all 1mH inductors) directly next to a switch. In the absence of a flyback diode, the circuit risked damaging the switch and also caused unwanted effects in the output voltage. The flyback diode had to switch at a similar rate to the switching rate of the buck converter's switch. To satisfy this design requirement, D1 was chosen to be identical to the diode from the buck converter, (15SQ100TR). It does not have to handle nearly as much current, but this saved design time by eliminating the need to spec out an entirely new part. The damping resistor was chosen somewhat arbitrarily, with the loose requirement of it being at least an order of magnitude smaller than the buck converter transistor's off-state resistance. The final implementation of the filter network can be seen in Figure 3.18.

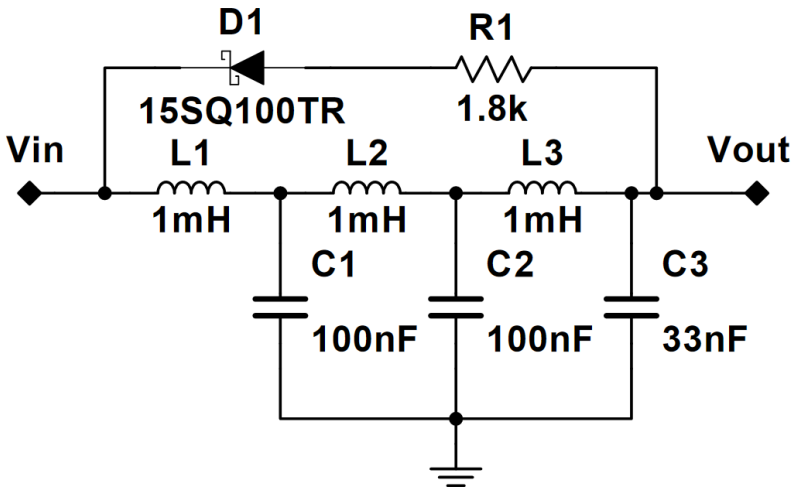


Figure 3.18: Final Filter Circuit Diagram

The inductance and capacitance values have been rounded to the nearest available component. The inductors are 1mH toroidal wire-wound inductors, while C1 and C2 are 100nF capacitors and C3 is a 33nF capacitor. In the final implementation of this circuit, some problems that had not previously occurred were observed. Instead of filtering, the circuit seemed to amplify the switching transients to about 60Vpk-pk, which exceeded the rating of the MOSFET driver. A similar, though less extreme behavior is shown in Figure 5.3 on page 37. A temporary solution to this was the addition of a 40V Zener diode across the MOSFET driver, seen in Figure 3.16 on page 21. A more permanent solution would be to change the filter design to one that does not rely on large inductors, but there was not time remaining in the project to implement this. For the purpose of testing the rest of the MPPT, the filter circuit was bypassed.

Current Sense Circuit

The current sensing circuit is the only circuit that had problems manifest only after the printed circuit board was fabricated. An oscillation in the amplifier that had not occurred in the breadboard prototype was found on the output of the amplifier. This oscillation is shown in Figure 3.19.

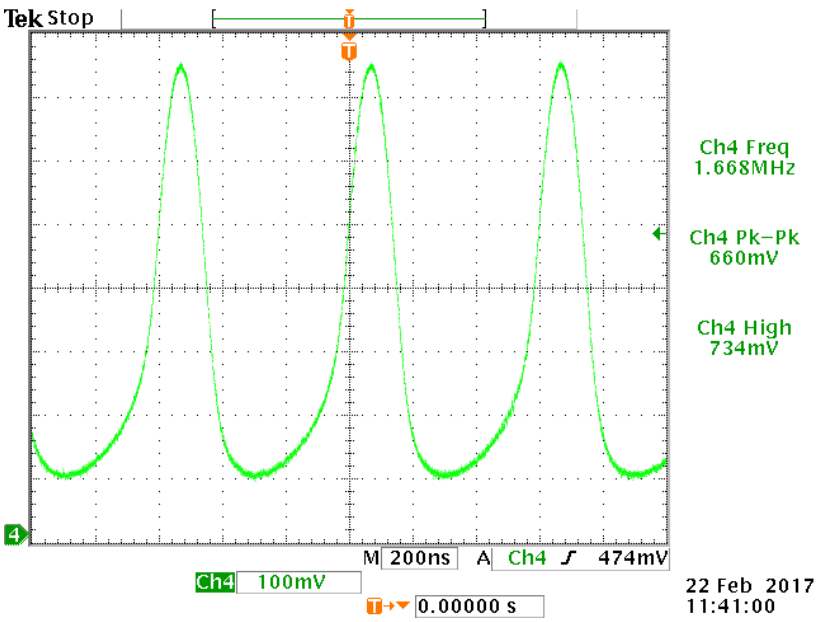


Figure 3.19: 2MHz Oscillation Between Amplifier and MCU

This was solved by adding a 10pF capacitor to the feedback network, as seen in Figure 3.20 on the following page, speeding up the amplifier's response to the oscillation and cancelling it. The results of this change are recorded in Figure 3.21.

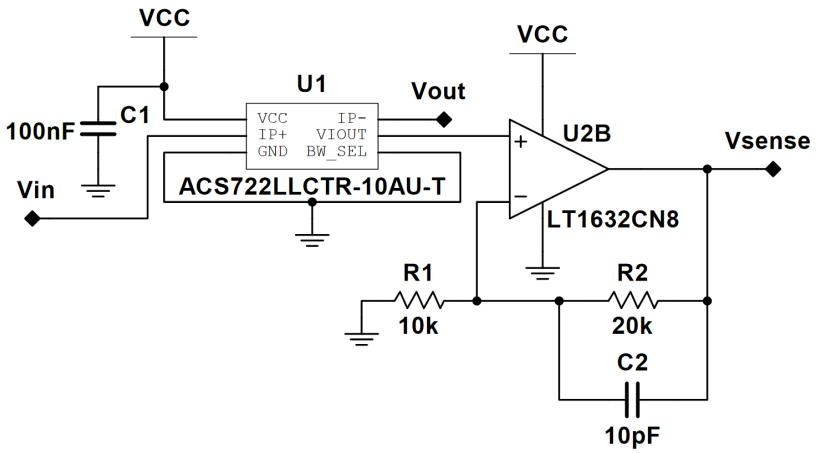


Figure 3.20: Final Current Sensor Circuit Diagram

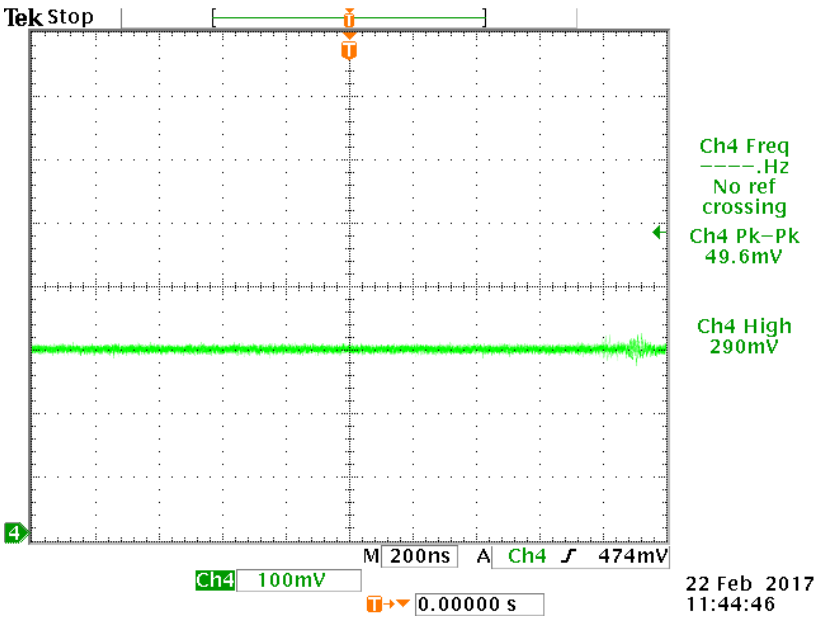


Figure 3.21: Reduced 2MHz Oscillation Between Amplifier and MCU

Control Circuitry

The control circuitry underwent no changes between theory and practical implementation, its schematic is included here for completeness. There were, however, some problems with the linear regulator. The chosen linear regulator had a dropout voltage of 2V. This meant that the 5V supply used by the project fell within the dropout voltage of the regulator. A temporary solution was to increase the supply voltage, but for completeness, the part was changed to the ST Microelectronics LE33CZ-TR linear regulator, which has a maximum dropout voltage of 0.5V and will not cause the same problems with a 5V supply. During this replacement, the team discovered that the device footprint pin mapping they had made in NI Multisim was backward from the one the manufacturer specified. With these changes, the circuit supplied 3.24V.

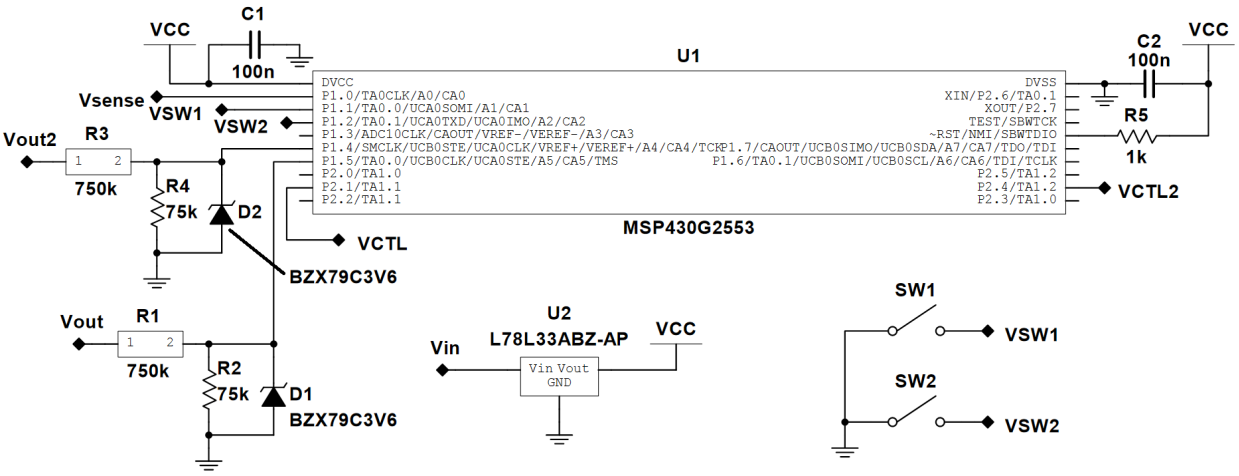


Figure 3.22: Final Control Circuit Diagram

Printed Circuit Board

The final circuits, excluding the solar cell emulator, were implemented on a printed circuit board. The board was fabricated by Advanced Circuits and designed to meet their standard specification. Full schematics used for pcb generation and rendered gerber file output can be found in Appendix A on page A-1 and Appendix C on page A-7 respectively. The only problem with the final implementation of the printed circuit board was the oscillation in the current sense circuit, the solution to which is covered in Section 3.3 on page 23. Figure 3.23 shows a picture of the final implementation of the Maximum Power Point Tracker on its printed circuit board.

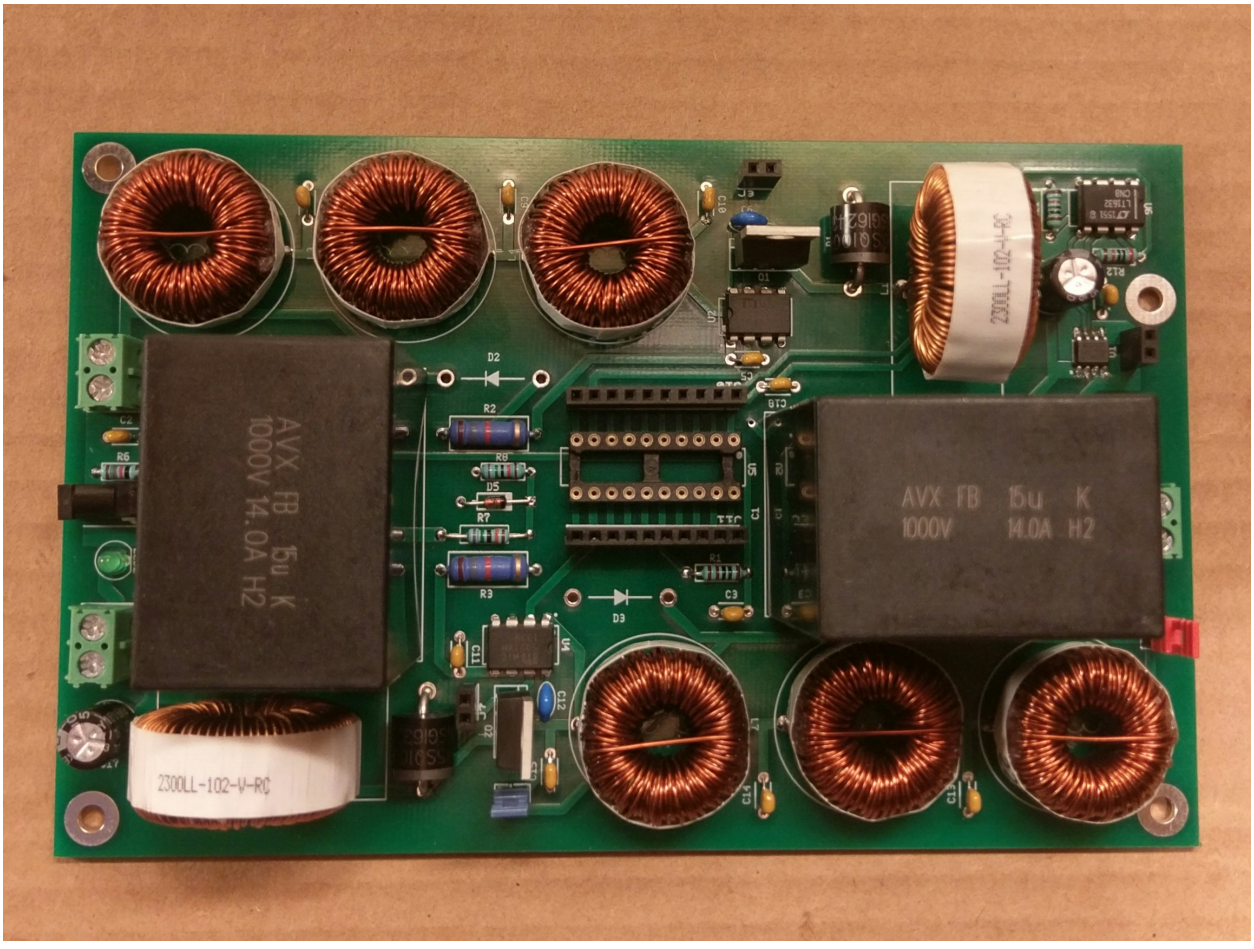


Figure 3.23: PCB Implementation of Final Circuit

4 Software Design

The software for the MPPT tracker was designed to control the two buck converters on the test platform. It had to support multiple Maximum Power Point Tracking algorithms, while handling data collection as well as control of the second buck converter. To meet these requirements, functionality was implemented in separate functional blocks to reduce dependencies between the two. This design allows a future user to easily implement and test additional MPPT algorithms.

4.1 Theory

Three Maximum Power Point Tracking algorithms were investigated in this project: Sweep, Perturb and Observe, and a Beta algorithm. Each of these algorithms have trade-offs in terms of efficiency and implementation complexity that warrant investigation.

The **Sweep Algorithm** is a simple algorithm for finding the maximum power point of a solar panel. The primary function of this algorithm is the sweep, where the duty cycle of a buck converter is incremented ("swept") from 0% to 100%. At each increment, the power is compared to the previous highest power, and if it has increased then the duty cycle is saved and the maximum power updated. When the sweep is complete, the duty cycle corresponding to the maximum power point for the conditions is known. After a set period of time, another sweep is performed to account for changing conditions. As changing conditions between sweeps can affect the amount of power delivered to the load, the time between sweeps will have an effect on the overall efficiency of this algorithm.

The **Perturb and Observe Algorithm** (Hill Climb Algorithm) is another simple algorithm that finds and tracks the maximum power point. This algorithm generally starts at a given duty cycle and then proceeds by increasing the duty cycle. After each adjustment it compares the power to the power from the previous adjustment, if the power has increased then it continues adjusting the duty cycle in the same direction. If the power decreases from one point to the next it will change direction. This algorithm continues to adjust the duty cycle in the direction of higher power, effectively "climbing the hill" of the solar panel's power vs. voltage curve. Once the maximum power point is reached, the algorithm will oscillate back and forth over it, which somewhat reduces the efficiency. In comparison to the Sweep algorithm, Perturb and Observe continuously tracks the maximum power point, increasing the overall efficiency.

The **Beta Algorithm** is a more complex algorithm that uses an intermediate variable β to help in calculating the new duty cycle. The algorithm requires 4 constants to be determined before the algorithm can run effectively. Two of these constants, β_{min} and β_{max} are calculated empirically and differ between solar panels. These values represent the β value of the solar panel at extreme temperature and luminance conditions. A third constant β_{g} is also chosen

to represent the ideal maximum power point and is used when finding error in this algorithm. β_{g} is often chosen as the midpoint between $\beta_{\min}$ and $\beta_{\max}$ as this should be close to the actual maximum power point.

Each cycle of this algorithm starts by calculating the value β with the following equation:

$$\beta = ln(\frac{I_{pv}}{V_{pv}}) - c * V_{pv} \tag{4.1}$$

Where c is a solar panel specific constant defined by:

$$c = \frac{q}{(N_S * A * K * T)} \tag{4.2}$$

Where q is the charge of an electron, $\mathrm{N_s}$ is the number of solar cells in the solar panel, A is the quality factor of the junction, K is the Boltzmann constant, and T is the temperature of the panel which is usually taken as 25°C. If the calculated β value is within the range $\beta_{\min}$ to $\beta_{\max}$ the algorithm switches to another method to find the actual maximum power point. The previously discussed Perturb and Observe algorithm is commonly used in applications of this algorithm. If the current value is outside of this range then the algorithm continues by calculating the change in duty cycle based on the error in the current value from the ideal β value as seen in:

$$\Delta D = N * (\beta_a - \beta_g) \tag{4.3}$$

N is a multiplier of the error in the algorithm and must be tuned to obtain the desired response. The duty cycle is then updated based on this value and the algorithm repeats.

Some work has been done in altering the algorithm to automatically calculate the proportional constant N. This auto-tuning algorithm calculates a new value N whenever the value moves out of the range $\beta_{\min}$ to $\beta_{\max}$. N is calculated from the Maximum duty cycle, the previous and current values of β with the following algorithm:

$$N = \frac{\Delta D_{max}}{(\beta_a(k) - \beta_g)} \tag{4.4}$$

While this adds complexity to the overall algorithm, this modification removes a step in the tuning process and improves overall efficiency of this algorithm.

4.2 Simulation

Before implementing the algorithms discussed above, it was important to simulate the behavior using sample data. The simulation would accept samples of the voltage and current from a solar panel and would then emulate the performance of an algorithm for a set period of time with the given data. As this simulation requires data captured from a solar panel it's overall utility is limited to analyzing the sweep algorithm. The simulation was made in MATLAB and the script is available in Appendix G.

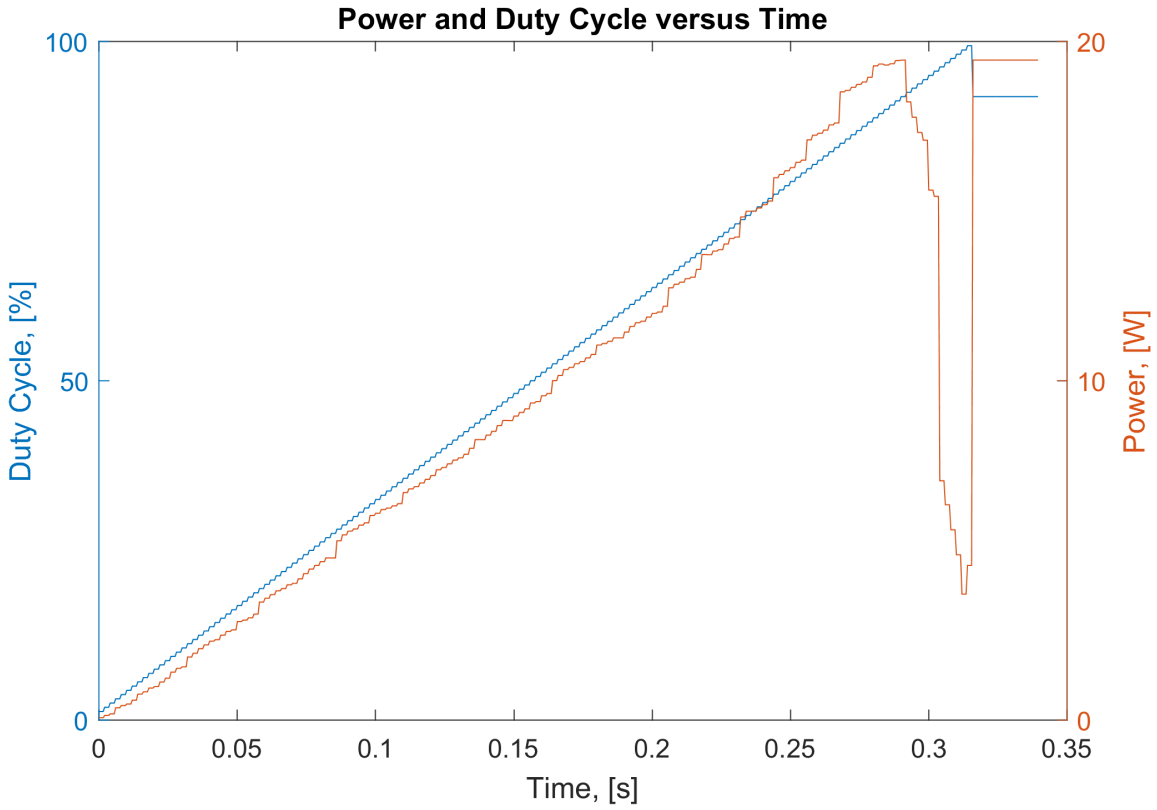


Figure 4.1: Simulation of Sweep Algorithm Finding Maximum Power Point

Analysis of the other algorithms mentioned requires the simulation of changing solar conditions, and a fundamentally different approach to building the simulation. As the physical implementation of the perturb and observe algorithm was functional at this time, continuation of the simulation was deemed outside of the scope of the project and was not pursued further. The simulation was limited to the Sweep algorithm which simply requires data from a sweep of the IV curve of the solar panel, as seen in Figure 4.1.

4.3 Implementation

The control algorithms discussed above must be implemented on the Texas Instruments MSP430 microcontroller selected for use. More information about this microcontroller and the control circuitry that goes with it can be found in Section 3.1 on page 12 and Section 3.3 on page 25. This microcontroller has a number of features that make it useful for this application, including a large amount of configurability of its various peripherals. Each of these peripherals has multiple options that must all be carefully chosen to work together correctly. The software must configure each of the required peripherals as well as implement data collection, and control algorithms for the two separate buck converters.

The internal clock is the first thing to be configured on the MSP430. It has been configured to run at the maximum clock speed of 16MHz and is used to provide timing signals to most of the peripherals including Timer 1. Timer 1 generates the PWM signal used to control the two buck converters. The timer is controlled by 3 registers, TA1CCR0, TA1CCR1, and TA1CCR2. TA1CCR1 and TA1CCR2 are configured to disable two GPIO pins, and are used to control the duty cycle of the PWM signal. TA1CCR0 is used to enable both GPIO pins and determines the overall clock speed of the PWM signal. The required value for the TA1CCR0 register can be calculated based on the desired PWM frequency and clock frequency using

$$R_{val} = \frac{F_{clk}}{F_{pwm}} = \frac{16MHz}{50kHz} = 320 \tag{4.5}$$

This also determines the precision of the duty cycle used in the buck converter control algorithms. The duty cycle is then controlled by setting one of the other registers to a value between 0 and 320. These registers will maintain their value, meaning that the PWM signal will stay at the same duty cycle until it is updated without requiring any input from the processor.

An auxiliary clock signal generated by a very low frequency oscillator is configured to run at 12kHz. Timer 0 uses this clock signal to provide timing for data collection. This timer is configured to generate an interrupt when the timer counts to 3, which results in an interrupt occurring at a rate of 4kHz.

The MSP430 has a number of General Purpose Input/Output (GPIO) pins that are used to interface with the various sensors and outputs on the solar maximum power point tracker. These GPIO pins must be configured correctly before being used. This configuration includes setting the two pins used for PWM, P2.1 and P2.4, to output, and configuring them for their primary function, which in this case is Timer outputs. Appendix H.2 on page A-46 contains information about MSP430 pin functions. Two pins are connected to push buttons for testing purposes and must be configured as inputs with their corresponding pull-up resistors turned on. Finally, Texas Instruments recommends that any unused GPIO pins are configured as Outputs set to off to reduce stray power consumption. [14]

The 10-bit Analog to Digital Converter (ADC) is configured to measure V_{MPPT}, I_{MPPT}, and

V_{OUT}. The ADC uses V_{CC} and V_{SS} for the reference voltage as this allows for a large dynamic range, and the extra precision of using a different voltage reference is not necessary. Each of the input pins, A0, A4, and A5 are configured as analog inputs, which configures them for usage with the internal ADC. The ADC clock is configured to operate at MCLK divided by 4, for a 4MHz clock signal. The sample & hold time is configured to be 64x this clock signal to ensure that the readings of the high impedance voltage signals would be accurate. These options provide a Sample and Hold time of 16µsec, which meets the minimum sample time as shown below.

$$t_{Sample} > (R_s + 2\text{k}\Omega) * 7.625 * 27\text{pF} \tag{4.6}$$

Where R_s is given by $R_1 \| R_2$, as below:

$$R_s = \frac{750\text{k}\Omega * 75\text{k}\Omega}{750\text{k}\Omega + 75\text{k}\Omega} = 68.18\text{k}\Omega$$

This gives:

$$t_{Sample} > (68.18\text{k}\Omega + 2\text{k}\Omega) * 7.625 * 27\text{pF}$$

$$t_{Sample} > 14.4\mu\text{sec}$$

Data collection is an integral part of the software for the MSP430 as each of the control algorithms requires information about the status of the buck converters. Three different measurements are taken with the 10-bit ADC, the voltage after the power conditioning buck converter (V_{OUT}), the voltage (V_{MPPT}) and current (I_{MPPT}) after the MPPT buck converter. These three measurements are taken sequentially, starting with V_{OUT}, then V_{MPPT}, and finally I_{MPPT}. The measurement process is started by the Timer0 Interrupt Service Routine (ISR) which runs at approximately 4kHz. The Timer 0 ISR configures the ADC to measure V_{OUT} and then starts a capture. When the ADC has finished capturing this value the ADC ISR is called to store the measurements. The ADC ISR is also responsible for starting collection of the next piece of data, as well as enabling the control algorithms for the output buck converter and the MPPT buck converter as necessary. To alleviate any adverse effects caused by noise in the measurements, the software calculates the average over 8 samples for each measurement. The control algorithms are called only after 8 measurements have been taken, which reduces the processing required while maintaining a reasonable control speed.

The power conditioning buck converter is designed to maintain a stable output voltage even as the MPPT voltage varies. As the load on the MPPT may be sensitive to extreme fluctuations in voltage it is important that the control algorithm be able to quickly respond to any fluctuations in output voltage. To achieve this goal, a Proportional-Integral (PI) loop was implemented. This control loop can be tuned to provide a very quick response to large transients without oscillation during steady state operation. As the MSP430 microcontroller does not have a Floating Point Unit (FPU) it was important that this implementation work with integer values. This algorithm must ensure that the integral component does not overflow. The algorithm uses the last 16 bits of a 32 bit integrator to improve dynamic range while also allowing small amounts of error to accumulate and eventually be corrected. For the purposes of this project the PI parameters were chosen experimentally as it was deemed unnecessary to perform a full mathematical analysis of the system response.

The main focus of the software is to support the usage and testing of multiple maximum power point tracking algorithms. Support for multiple MPPT algorithms can be compiled into the software and loaded on the MSP430. The algorithm being used can then be changed using an auxiliary button. This makes it easy to demonstrate the functionality of multiple algorithms and compare their performance. For many of these algorithms there are two variables that affect the efficiency, step size and algorithm speed. The algorithm speed is mainly limited by the transient response of the buck converter, as there must be enough time for the circuit to settle before taking new measurements. The step size depends on the algorithm being used and must be chosen to balance speed and precision.

The sweep algorithm was the first algorithm to be implemented due to its simplicity. This algorithm starts at a 20% duty cycle and sweeps to 100% before setting the maximum power duty cycle. A starting point of 20% was chosen to improve the efficiency of the algorithm by reducing the time spent sweeping. The duty cycle is controlled by setting the TA1CCR1 register to a value between 0 and 320 as discussed in Section 4.1. As the amount of time necessary to perform a sweep is directly correlated with the step size used and the amount of time at each step it is important to choose appropriate values for both of these variables. The step size is limited to 0.3125% due to the configuration of the timer, as well as the need for a fast sweep. It was determined that a step size of 2 increments, or 0.625% would provide appropriate balance between accuracy and speed. The time it takes to complete a sweep can be calculated using the following equation:

$$T = \left(\frac{D_E - D_S}{D_{step}}\right)\left(\frac{1}{F_s}\right) \tag{4.7}$$

Where D_e is the ending duty cycle, D_s is the starting duty cycle, D_{step} is the duty cycle step size, and F_s is the algorithm frequency. Sweeping from 20% to 100% using a step size of 0.625% duty cycle and a frequency of 25Hz as used in testing, the sweep would take 5.12 seconds to complete. As D_{step} is already quite low, the best way to improve the sweep time is to increase the algorithm frequency F_s. As it is currently quite low at 25Hz, this can be increased dramatically while maintaining accuracy.

The perturb and observe algorithm was the second algorithm implemented. The implementation for this algorithm is straightforward and follows the theory described previously. In this implementation the duty cycle is initialized at 25% as it reduces the time it takes to reach the maximum power point by skipping the low efficiency region of the buck converter. To reduce the effects of noise in the measurements the duty cycle is only adjusted if the current power varies by a predetermined threshold. The algorithm then adjusts the duty cycle with a preset step size based on the change in power from the previous step. If the power was increased by the previous adjustment then the next adjustment continues in the same direction, otherwise the direction is changed.

The Beta algorithm was the third algorithm to be implemented. As discussed in the theory section, this algorithm finds an intermediate variable β from the current and voltage of the MPPT buck converter. If the β value is outside of the range β_{min} to β_{max} then the algorithm continues calculate a new step size N, before calculating the new duty cycle. If the β value is

inside the range then the perturb and observe algorithm is used to find the maximum power point for the given conditions. This algorithm has a number of constants that must be determined empirically, β_{min}, β_{max} and β_{g} Each of these values must be found for a specific solar panel.

The Beta algorithm is more efficient than the previous two algorithms, with the trade-off that it requires more processing power. The algorithm requires the usage of floating point math operations that require advanced features on a microcontroller. Unfortunately the microcontroller that we have chosen for this project does not have a Floating Point Unit (FPU) or a hardware multiplier. This means that the C code compiler must convert these operations to assembly instructions that the MSP430 can perform. Implementing these instructions in assembly requires a significant number of operations which requires far more processing time than if they were implemented in hardware. This reality prevented the full implementation and testing of the Beta algorithm for this report. Due to the large number of options in the MSP430 line, a different microcontroller with an FPU and hardware multiplication support could be used with little modification to the software, but a fairly large hardware change would be required. Due to time constraints, the Beta algorithm did not undergo testing.

5 Experimental Results

This chapter discusses the results of the testing performed using the circuits built for this project. This includes the performance of the circuits themselves, the performance of software algorithms relative to each other, and the combined performance of both hardware and software.

5.1 Circuit Performance

This section covers the performance of the hardware platform without taking into account any effect software might have on the system. Hardware testing focused on reliability and input-to-output efficiency. Other factors, including ripple voltages and thermal performance were also considered.

Testing Criteria

The Solar Cell Emulator's tests were used to determine its ability to function as a test fixture, rather than to prove its dynamic performance. Testing was accomplished using a MightyWatt 70W electronic load [15] using a current-controlled linear sweep from open circuit load to short circuit load conditions in order to gather voltage data to build the current-voltage characteristic of the emulator circuit.

The input filter was tested by measuring the voltage at each stage of the filter. A fixed input voltage and buck converter duty cycle were used to ensure stable conditions. The output of the buck converter was also tested to ensure that the ripple voltage was within the design constraints. This test was performed with a constant input voltage of 10V, duty cycle of 80% and a load of 8Ω.

The real world efficiency differences are best seen by measuring the energy provided to a static load by each algorithm. This test is performed using a fixed 8 Ω load, and the solar cell emulator connected to a fixed voltage power supply.

Solar Cell Emulator

The solar cell emulator circuit designed for this project was crucial to the accurate testing of the MPPT tracking performance. This made it important to verify the behavior of the solar cell emulator. The current and voltage was collected at regular intervals using the MightyWatt software while varying the load. Data was collected for each of the two characteristic settings of the Solar Cell Emulator test circuit to show that it produced unique I-V characteristics. The current and power for each setting can be seen in Figure 5.1. The first characteristic has been configured to emulate a smaller radiance, with a maximum power of 16.6W, whereas the second characteristic has a maximum power of 25.6W. The current and voltage data collected during testing was then used in simulations of the sweep algorithm discussed in Section 4.2. Using this data allows for a more accurate simulation result.

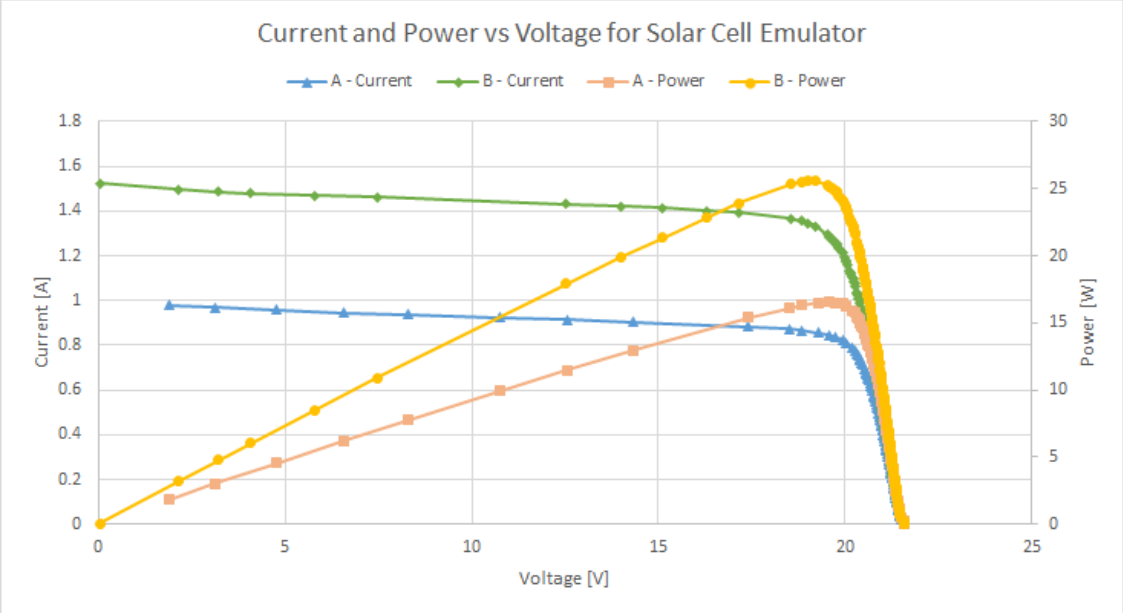


Figure 5.1: Solar Cell Emulator I-V and P-V Characteristics

The solar cell emulator relied on an IRF5920 P-channel MOSFET. This transistor got hot enough during initial testing to warrant the installation of a heat sink and fan. Due to lack of available testing equipment[1], the exact thermal performance of the device is unknown, but with the heat sink and fan it stayed at approximately 25°C for even extended run times.

MPPT Performance

Before the team was able to test the software control algorithms, the functionality and efficiency of the Maximum Power Point Tracking hardware had to be evaluated. Each functional block of the hardware was tested to verify that it functioned correctly and meet the performance criteria.

[1]Unfortunately this data is qualitative, as it was taken with a human finger rather than a thermometer.

The filter was shown to operate as expected during breadboard testing by reducing the voltage ripple present at each stage of the filter. The results of this test can be seen in Figure 5.2. The ripple directly from the buck converter is seen on Channel 3, the input after going through stage 3 is shown on Channel 2, and the solar panel input is shown on Channel 4. The ripple voltage at the solar cell emulator is 304mV, having been reduced from a ripple voltage of 5.96V. The resulting ripple voltage meets the requirement to provide a stable load to the solar panel.

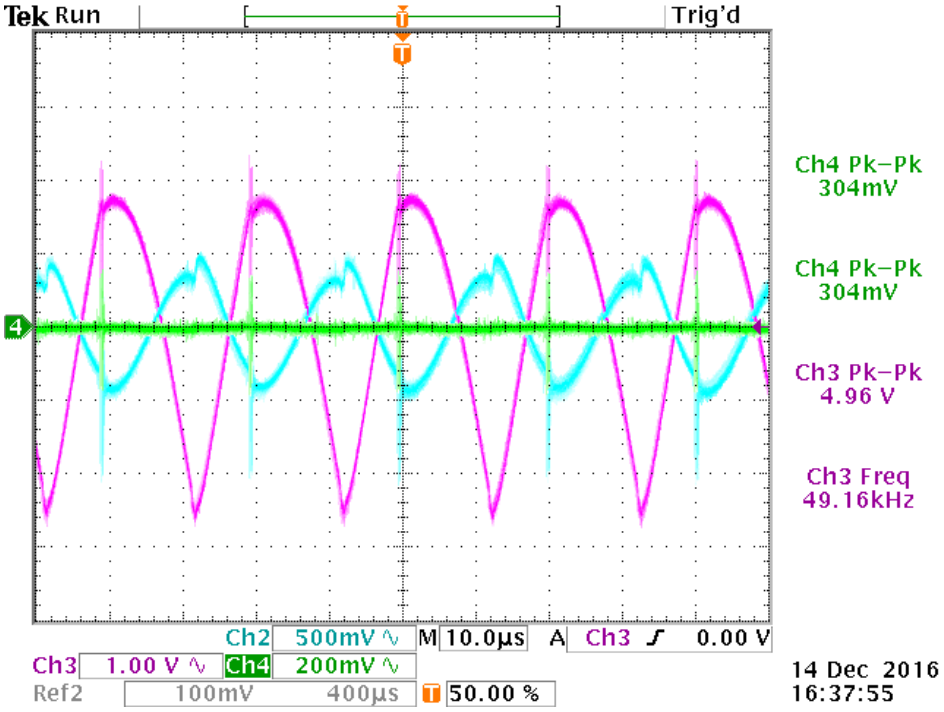


Figure 5.2: Breadboard Filter Test Results

Unfortunately, the results obtained during breadboard testing were not indicative of the performance of the final filter. Instead of reducing the amplitude of the voltage ripple, the filter seemed to amplify it, to the point of exceeding the rating of the MOSFET driver and burning out the chip. This behavior was only observed in the PCB version of the circuit, indicating that the breadboard filter was somehow different electrically. This is discussed in more depth in Section 3.3 on page 23. The test results are shown in Figure 5.3 on the following page, where Chanel 1 is the input to the filter and the channels progress sequentially between filter stages to Channel 4 which measured the source pin of the MOSFET of the buck converter.

The output ripple from the buck converter was tested to ensure that it fell within the desired operating conditions from Section 3.1 on page 9. The ripple voltage from the output can be seen on Channel 4 in Figure 5.4 on the following page. The ripple voltage was found to be 140mV which is below the desired ripple voltage of 500mV.

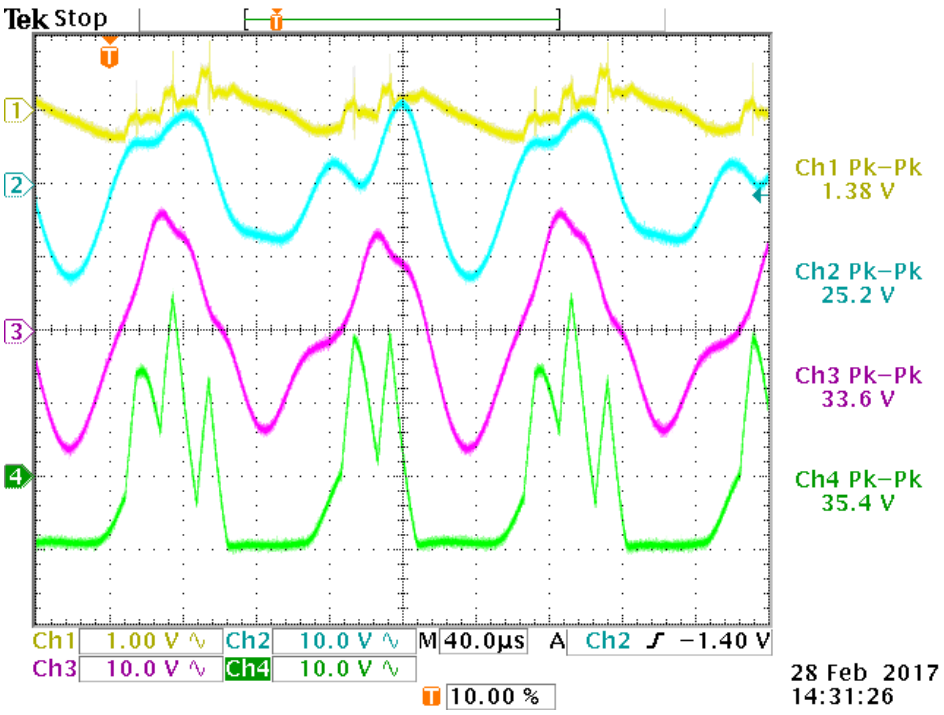


Figure 5.3: PCB Filter Test Results

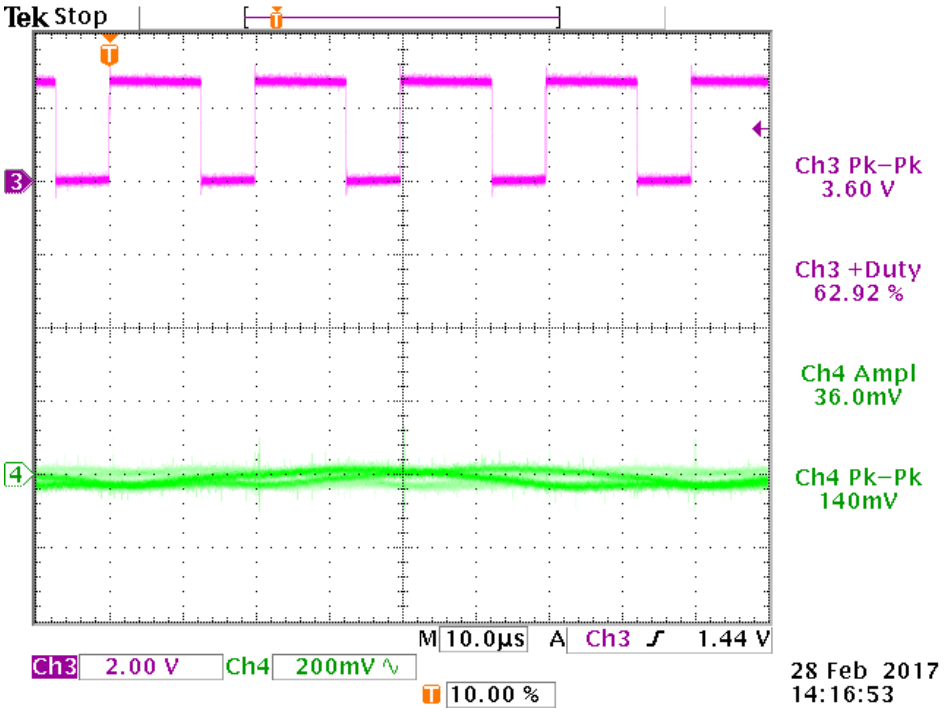


Figure 5.4: Buck Converter Output Voltage Ripple

The output power versus the input power, or efficiency, of the buck converter was measured at different duty cycles. The efficiency was tested for duty cycles between 20% and 80% at 20% intervals shown in Figure 5.5. The efficiency was found to range from 32% to a high of 79% efficiency at an 80% duty cycle. The relative efficiency of different software algorithms was the focus of this project, so the relatively low efficiency of the buck converter was not considered an issue.

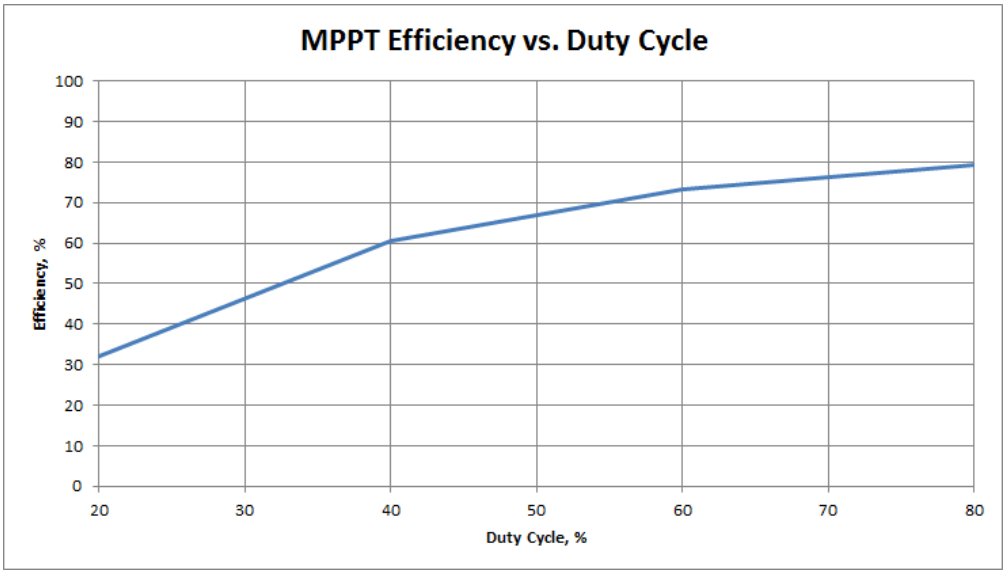


Figure 5.5: Buck Converter Efficiency

The main source of energy loss in the maximum power point tracker was the DMT6009 MOSFET switch. This transistor got hot enough during initial testing to warrant the installation of a heat sink and fan, though not quite as hot as the transistor in the solar cell emulator[2]. Even with the installation of the heatsink and fan the transistor tended to heat up after extended run times, but only to an estimated 40°C, or slightly higher than human body temperature.

[2]Unfortunately this data is qualitative, as it was taken with a human finger rather than a thermometer.

5.2 Relative Software Performance

This section discusses the performance of software algorithms relative to each other, without taking into account any effects of the hardware platform. The algorithms are compared based on their initial time to reach the maximum power point, and their step response. By using the same hardware platform to test multiple maximum power point tracking algorithms it is possible to compare the efficiencies of each algorithm. Such a comparison should eliminate any differences caused by the underlying hardware and allow for a direct comparison of MPPT algorithms.

Sweep Algorithm

The time it takes for the sweep algorithm to first reach the maximum power point is dependent upon the step size and the frequency it adjusts at. As these parameters are hardcoded this time can be calculated before hand to ensure that it is appropriate. Given an update frequency of 50Hz and a step size of 0.625%, it should take 3.2 seconds to complete the sweep and settle on the maximum power point. The initial sweep performed in a test run can be seen in Figure 5.6. Cursors on the oscilloscope show that the sweep took approximately 3.16 seconds, the difference with the calculated time can be accounted for by inaccuracies in cursor placement.

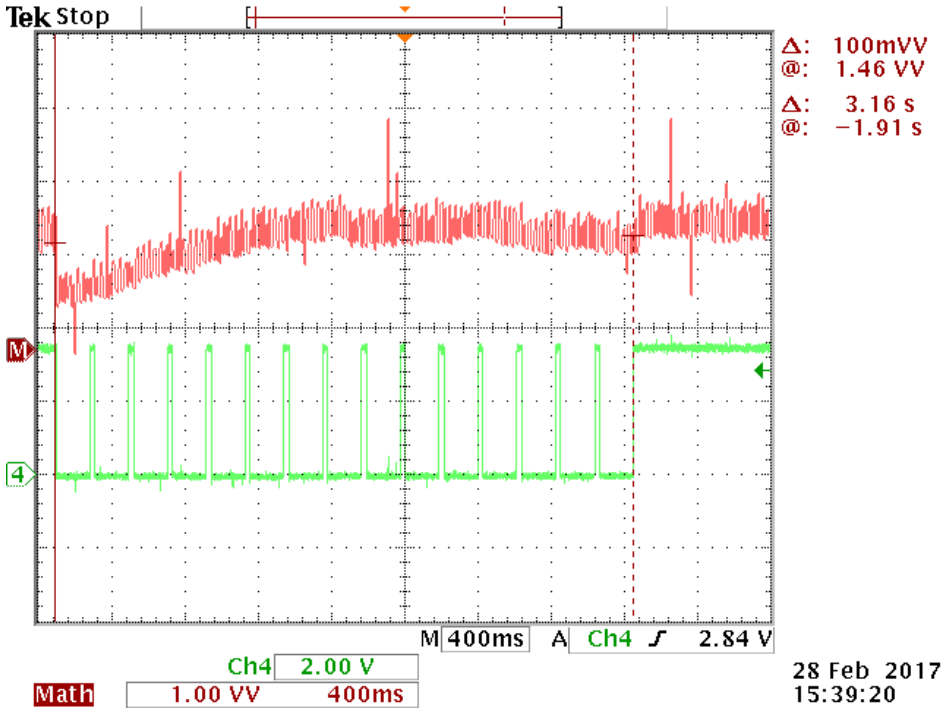


Figure 5.6: The Sweep Algorithm Testing the Power Between 25% and 100% Duty Cycle

The relative power is shown in red and is graphed using the multiplication functionality of the scope based on readings from the current and voltage inputs to the MSP430. As these

inputs are not scaled, and the current measurement is not zero when zero current is passing the power value is only useful for relative comparisons. Channel 4 gives an indication of the duty cycle, as it is driven high at 5% increments in the duty cycle. It can be seen in this figure that the power starts to decline as the duty cycle approaches 100%, and then increases to the maximum as the algorithm finishes the sweep.

The sweep algorithm relies upon short periods to perform a complete sweep of the duty cycle before settling on the maximum power point. During these periods the output power will decrease to zero and slowly increase to the maximum. The desire for infrequent sweeps is offset by changing solar conditions which will affect the maximum power point. It is thus important to balance the frequency of sweeps with the rate of change in luminance. The frequency of sweeps would need to tuned for the specific solar conditions experienced. For testing purposes a small sweep interval of approximately 5 seconds was used to allow for data collection.

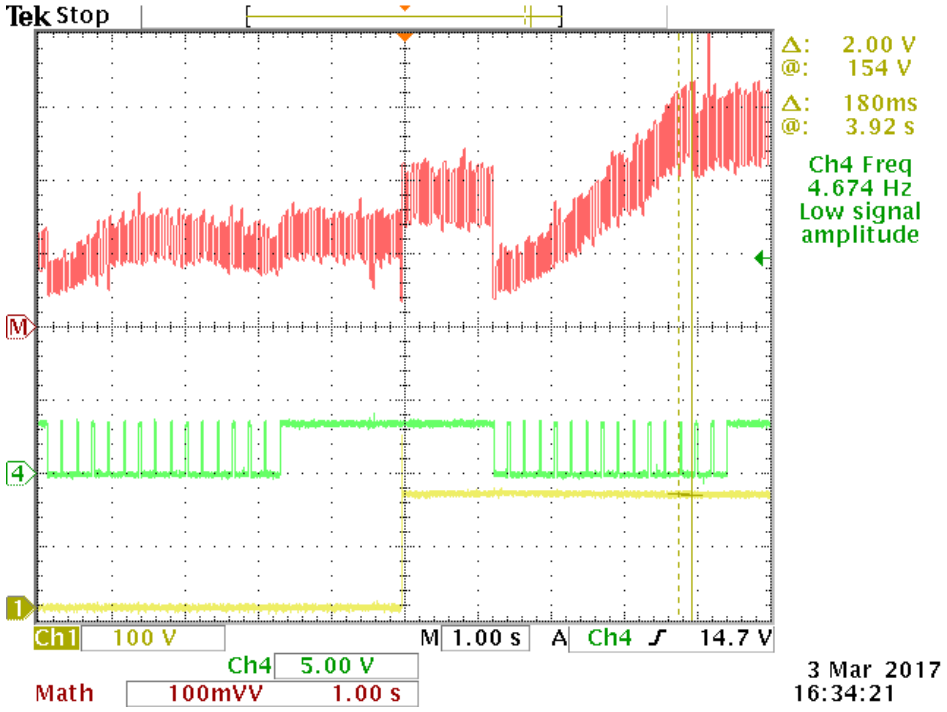


Figure 5.7: The Sweep Algorithm Reacting to Changing Solar Conditions

Figure 5.7 shows the initial sweep, along with a second sweep performed after the solar conditions have changed. Channel 1 represents the solar cell emulator characteristic being used. A low signal represents the first characteristic, and a high signal represents the second characteristic. As in Figure 5.6, Channel 4 represents the duty cycle during a sweep, and the relative power is again shown in red. The sweep algorithm will generally perform better in the real world than indicated in this test. This is due to the relatively slow change in luminance under the sun. As actual solar conditions change slowly, the chosen power point will drift only slightly from the actual maximum power point.

Perturb and Observe

The perturb and observe algorithm attempts to reach the maximum power point by continuously adjusting the duty cycle and evaluating the change in power. By always moving in the direction of power increase this algorithm should eventually settle on the maximum power point. This algorithm will never actually stop on the maximum power point, instead it will oscillate around it. To reduce the size of these oscillations and improve the overall efficiency a small step size is desired. At the same time, a large step size is desired to reduce the time to reach the MPP. A balance between these two desires must be found to maximize the efficiency of this algorithm. In testing a step size of 0.625% was used.

The time it takes for the algorithm to first reach the MPP is measured in the same manner used for the sweep algorithm. That is, the duty cycle starts at 25%, with the circuit connected to the solar cell emulator, and a static load. The relative output power is displayed along with a secondary indicator in Figure 5.8. The secondary indicator shows the direction that the algorithm is moving in, with a low signal representing a decreasing duty cycle, while a high signal represents increasing duty cycle. This algorithm took 1.38 seconds to first reach the maximum power point. When the maximum power point is reached the direction indicator will rapidly switch back and forth, or in this case from low to high.

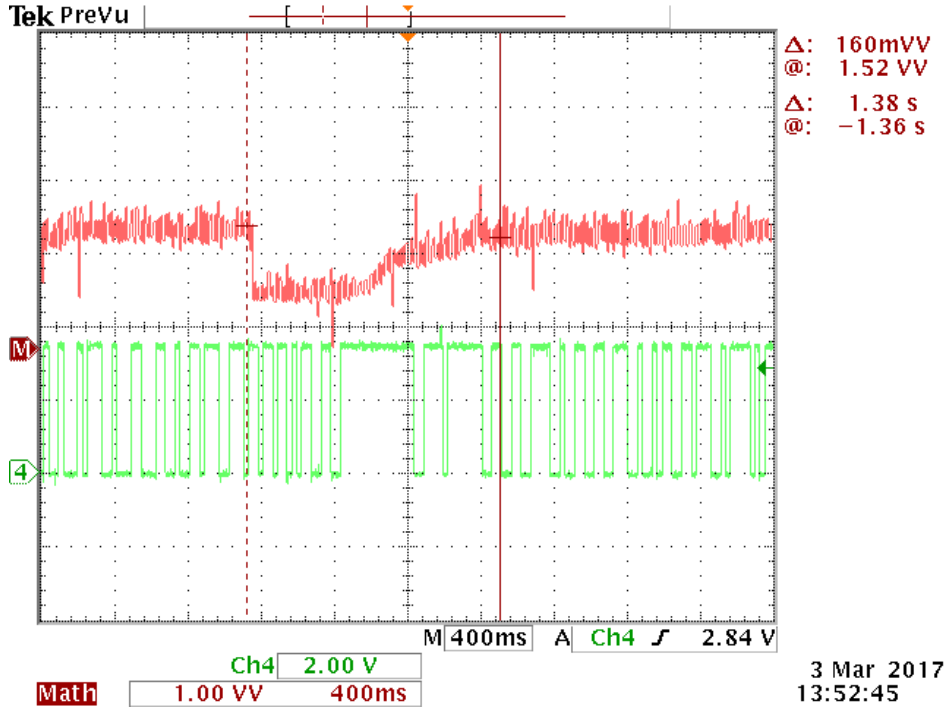


Figure 5.8: The Perturb and Observe Algorithm Immediately Following System Reset

Perturb and Observe is constantly moving about the maximum power point after first reaching it; this behavior should allow it to rapidly follow changing solar conditions. It should also be able to handle a sudden change in solar conditions, taking a small number of cycles to settle on the new MPP. The algorithm's response to a sudden change in solar conditions

is shown in Figure 5.9. It can be seen that the duty cycle decreases for a short period of time before it begins to increase. This behavior is expected in simple implementations of the Pertub and Observe algorithm as the power starts to increase when the duty cycle was moving downwards. This behavior can be corrected in software resulting in even further increases in performance, but was not implemented for this test. Eventually the duty cycle starts to increase, and rapidly increases the power delivered to the load until it reaches the maximum power. The algorithm reaches the new MPP in 1.3 seconds and quickly settles into a small oscillation as before.

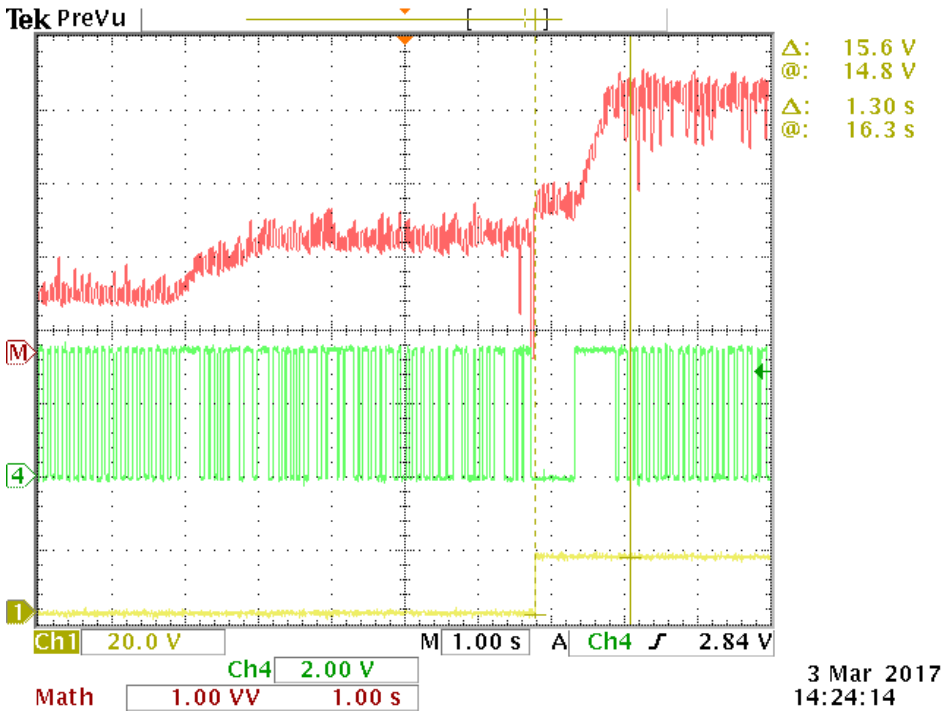


Figure 5.9: The Perturb and Observe Algorithm Reacting to Changing Solar Conditions

5.3 Combined Performance

This section reports the combined performance of all algorithms implemented on the hardware platform, final combined efficiencies under various tests and other considerations from systems integration that were not discussed in the relative performance sections. The efficiency of each algorithm relies upon its ability to quickly reach the maximum power point when conditions change. It was important to be able to respond to both small and large changes in solar conditions as they change gradually throughout the day, but can also experience sudden transitions.

Sweep Algorithm

The sweep algorithm was configured to sweep at 5 second intervals and was used to track the maximum power point for one minute. The sweep algorithm sweeps the entire I-V curve times throughout the testing period. The power delivered to the load over the testing period is shown in Figure 5.10[3]. The sweep algorithm delivered 372 Joules to the load over the testing period.

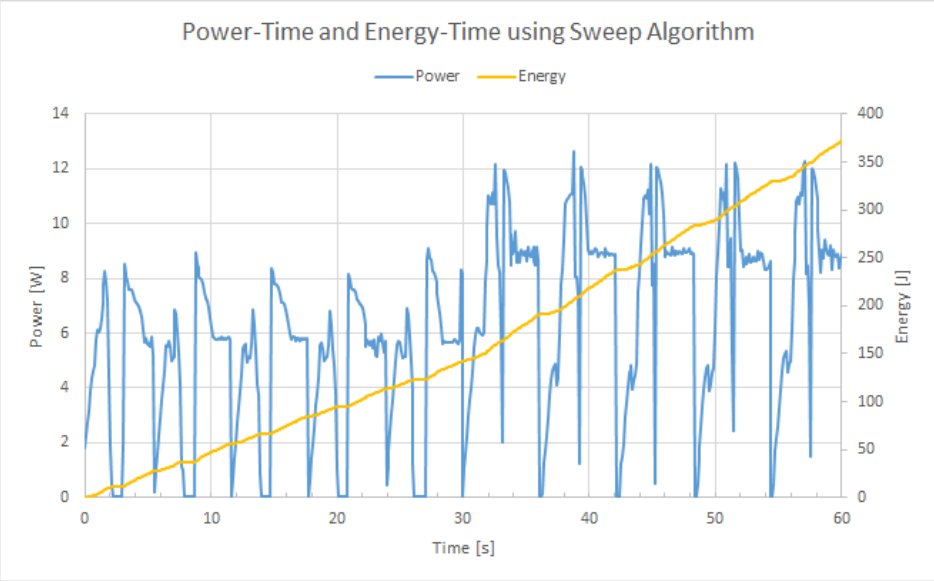


Figure 5.10: Power and Energy Delivered to Load by Sweep Algorithm

The power over time curve also provides information about the way the algorithm operates. For the sweep algorithm, each sweep of the I-V curve is visible as a dip in the power output highlighted in Figure 5.11.

The power drops to 0W at the beginning of the sweep, then increases to the maximum power point, during this sweep the power being delivered to the load is clearly non-optimal. For a sweep during the first solar characteristic, 9.5 Joules are delivered to the load, versus a theoretical 18 Joules[4] at the maximum power point. When the maximum power point tracking is used for long periods of time, frequent sweeps can result in a significant loss of potential energy. The efficiency of this algorithm is linked to the frequency and speed of sweeps. This example used a large number of sweeps in a short time period to emulate the behavior of this algorithm when tracking over a longer time period.

[3]The power characteristic can be seen declining to a new steady state immediately after the duty cycle is set to the maximum power point. This phenomena is believed to be caused by using the previously mentioned MightyWatt dynamic load as a measurement device. The device was configured in series with an 8Ω load, and configured to present no additional resistance. The dynamic nature of this device is believed to have caused the discrepancy in power measurements.

[4]The final steady state power was used for all calculations to provide a more accurate representation

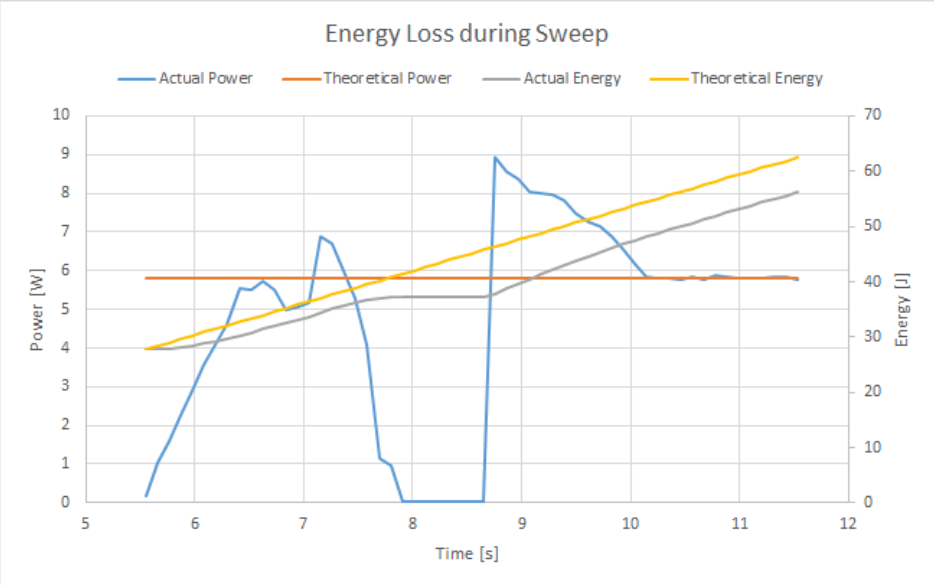


Figure 5.11: One Sweep and Hold Cycle with Theoretical Power and Energy

Perturb and Observe

The Perturb and Observe algorithm is run in the same configuration as before, with a 50Hz update frequency and a 0.625% step size. The power delivered to the load over the duration of the test is shown in Figure 5.12[5]. The algorithm immediately starts moving towards the MPP and reaches the maximum power point after approximately 1.2 seconds. The power then remains relatively constant until the solar cell emulator is switched to the other characteristic, where the power increases slightly before the algorithm adjusts to the new characteristic.

The algorithm then starts to adjust the duty cycle to reach the new MPP. During this transition the load is not receiving the maximum possible power, as seen by the dip in power before the algorithm settles upon the new maximum power point. This transition takes 5.44 seconds before reaching the new maximum power point. During this time, approximately 22 Joules of potential energy are lost while the algorithm seeks the new MPP. A total of 429 Joules are delivered to the load during the testing period.

[5]This testing was done in the same manner as used for the Sweep algorithm and faces the same issue with the power measurements

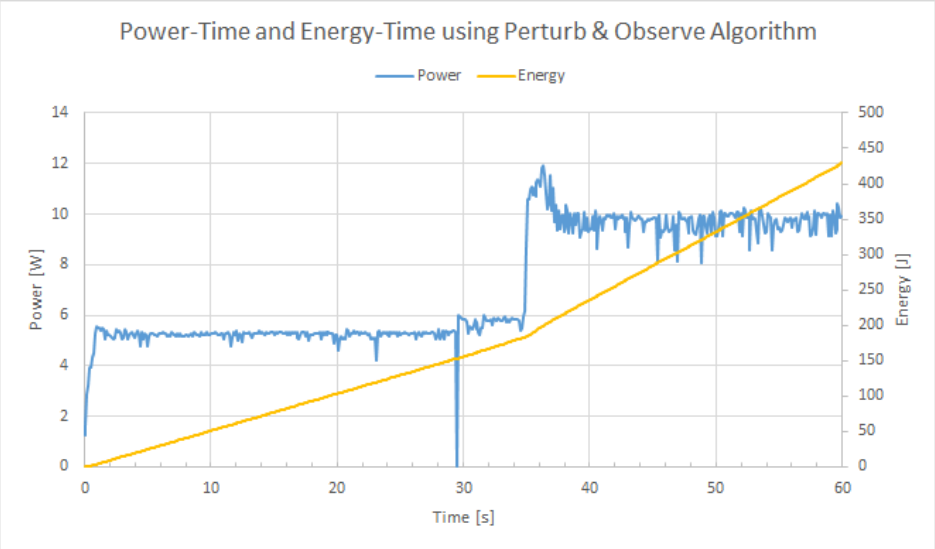


Figure 5.12: Power and Energy Delivered to Load by Perturb & Observe Algorithm

5.4 Summary of Results

Both the Solar Cell Emulator and the MPPT hardware performed adequately for their intended purpose. The filter circuit failed to perform as expected, but because the MPPT did not have to be connected to a physical solar panel, it could be bypassed without causing problems. The sweep algorithm works as designed, boasting high reliability if lower efficiency than the perturb and observe algorithm. The perturb and observe algorithm's more dynamic nature helps it respond faster to changes than the sweep algorithm, though it can occasionally get stuck on local maxima.

6 Conclusion

There are a number of topics that this project either started to investigate and had to abandon due to time constraints or appeared part-way through the project as relevant tangential inquiries. This section covers opportunities to build upon the work done in this project, as well as a summary of the major parts of the project.

6.1 Additional Work

This section covers parts of this project that were not pursued due to time constraints, tangentially related work that was outside the project's scope, and possible expansions upon this project.

Beta Method

The Beta Method for maximum power point tracking differs from the other types of maximum power point tracking discussed in this project in that it is an error-calculation method. This means it uses math to predict the location of the maximum power point from a set of known parameters of the solar array. An implementation of the Beta Method for an MSP430 was written as part of this project, but was not tuned or tested due to time constraints.

The Beta Method relies on precisely-tuned constants stored in the microcontroller to predict the location of the maximum power point. In order to correctly tune these constants, the MPPT needs to undergo testing, as these constants are experimentally determined and differ for different panels and MPPT hardware combinations. This requires an investment of time that unfortunately was unavailable during the course of this project.

The primary concern with the implementation of the Beta Method in the MSP430 is the reliance of the algorithm on a modulo operation. This is a floating-point division operation, and the MSP430 has no floating-point unit. The MCU already suffers slowdowns due to the need to handle 32-bit fixed-point numbers, multiplication and division with its 16-bit arithmetic logic unit, thus a microcontroller upgrade may be in order for implementing the Beta Method.

Advanced MCU Integration

This project's circuit board does not have an integral microcontroller. Instead, the board mounts a 20-pin DIP socket to fit an MSP430G2553 MCU. This is a low power and low performance microcontroller and certain parts of the maximum power point tracking algorithms implemented in this project could benefit from both faster clock speed and longer word size in processors.

Due to the advent of the Internet of Things, there is an increasing availability of high-power low-cost microcontrollers that offer some of the performance capabilities of full-size computers such as 32-bit words and faster clock speeds than regular microcontrollers with only slightly increased power draw and price. These MCU's also tend to boast features such as native ethernet support so that they can simply and easily interface with the internet. A possible upgrade to this project would be to replace the MSP430 with one of these more powerful microcontrollers and to take advantage of its features. Local or wide area network monitoring and control is one practical feature set that would be much more convenient to implement using one of these controllers. Other features include displays and graphics, decentralized coordination between multiple MPPTs, and intelligently determining to supply only the power required by the load or adapting to different circuit loads.

Self-Powering Apparatus: Battery and DC Transfer Switch Integration

Another intended feature of this project that was canceled due to lack of time was the ability of the MPPT to power its control circuitry from the solar energy it collected. Some framework for this already exists, though the DC transfer switch the system would have to rely on was not designed. While pursuing this goal, the addition of a battery to the system could be useful. These changes would reduce the amount of power the system would have to draw from the grid to power itself, theoretically making off-grid operation possible.

Solar Position Tracking Integration

This project developed a device to track the maximum power point of a photovoltaic. Many solar energy harvesters, however, are mounted on pivoting hardware so as to be able to point directly at the sun for the entire time it is out, increasing the harvested power. Since the MCU used for this project has spare I/O ports, a further expansion of this project could use those ports to control the azimuth and elevation angle of a solar panel to physically track the sun as well as the panel's maximum power point.

6.2 Concluding Remarks

This project investigated the efficiency of various maximum power point tracking algorithms on a single hardware platform. Additionally, it covered the development of a low-cost hardware platform with which to test these algorithms, the development of a low-cost solar cell emulator, and the implementation of the software algorithms to be tested. The circuits and the algorithms all worked as designed, except for the filter circuit. This project provides a hardware and software platform for future work to build upon and attempts to leave detailed resources for future projects to take advantage of.

References

[1] [Online]. Available: https://www1.eere.energy.gov/solar/pdfs/solar_timeline.pdf

[2] T. Barcelo, "Techniques to maximize solar panel power output," 2014. [Online]. Available: http://www.linear.com/solutions/4545

[3] A. Mukerjee and N. Dasgupta, "{DC} power supply used as photovoltaic simulator for testing {MPPT} algorithms," *Renewable Energy*, vol. 32, no. 4, pp. 587 – 592, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0960148106000644

[4] O. M. Midtgard, "A simple photovoltaic simulator for testing of power electronics," in *2007 European Conference on Power Electronics and Applications*, Sept 2007, pp. 1–10.

[5] T. Esram and P. L. Chapman, "Comparison of photovoltaic array maximum power point tracking techniques," *IEEE Transactions on Energy Conversion*, vol. 22, no. 2, pp. 439–449, 2007.

[6] P. Midya, P. T. Krein, R. J. Turnbull, R. Reppa, and J. Kimball, "Dynamic maximum power point tracker for photovoltaic applications," in *PESC Record. 27th Annual IEEE Power Electronics Specialists Conference*, vol. 2, Jun 1996, pp. 1710–1716 vol.2.

[7] S. Jain and V. Agarwal, "A new algorithm for rapid tracking of approximate maximum power point in photovoltaic systems," *IEEE Power Electronics Letters*, vol. 2, no. 1, pp. 16–19, March 2004.

[8] A. Ducimo, K. Sniekus, and N. Verlee, *Renewable Energy Applications*, ser. Major Qualifying Project. Worcester Polytechnic Institute, 2012. [Online]. Available: https://web.wpi.edu/Images/CMS/ECE/Roofttop_Solar_Panel_MQP.pdf

[9] A. C. Delphia and S. T. Veilleux, *Grid-Independent Charging Station with Power Flow Display*, ser. Major Qualifying Project. Worcester Polytechnic Institute, 2017. [Online]. Available: https://web.wpi.edu/Pubs/E-project/Available/E-project-121312-112925/unrestricted/Grid-Independent_Charging_Station_with_Power_Flow_Display.pdf

[10] A. W. Savoy, Giancarlo Sossavi, *Solar Charging Station*, ser. Major Qualifying Project. Worcester Polytechnic Institute, 2017. [Online]. Available: https://web.wpi.edu/Pubs/E-project/Available/E-project-030615-213336/unrestricted/MQP_Final.pdf

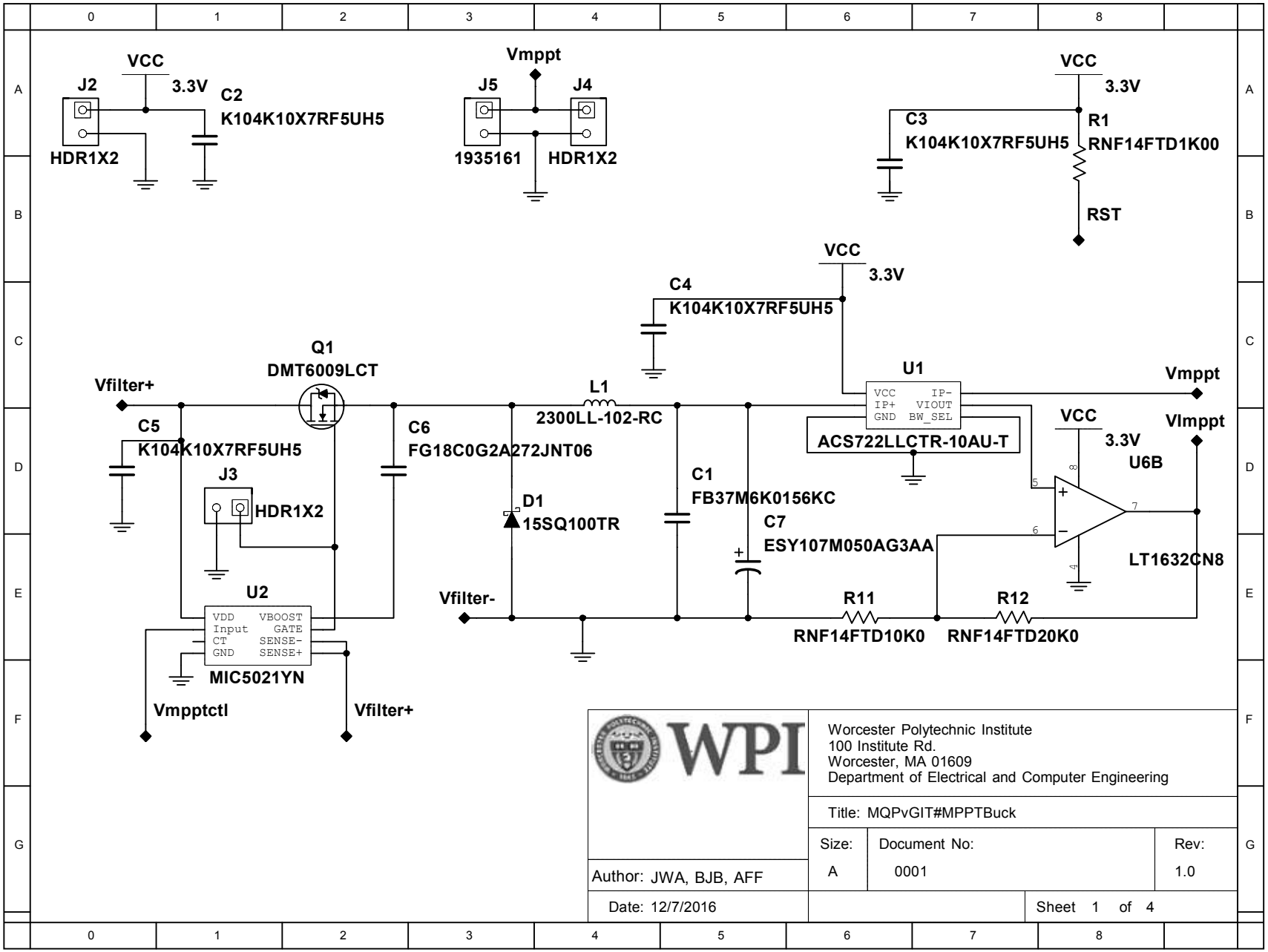
[11] H. Alberti and M. A. EGhani, *Maximum Power Point Tracker*, ser. Major Qualifying Project. Worcester Polytechnic Institute, 2014. [Online]. Available: https://web.wpi.edu/Pubs/E-project/Available/E-project-032114-223145/unrestricted/Maximum_Power_Point_Tracker.pdf

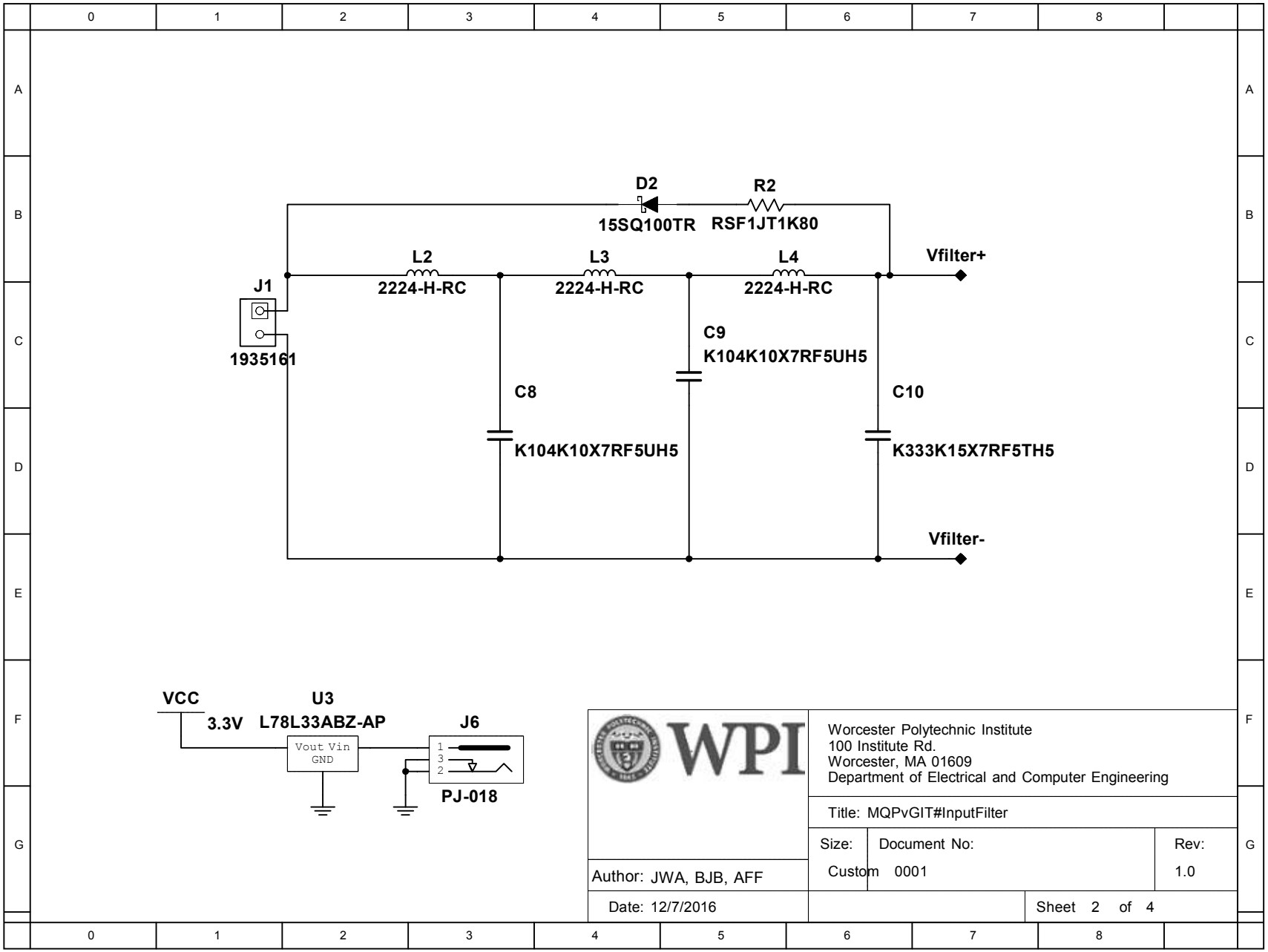
[12] D. W. Hart, *Power electronics*, 1st ed. McGraw-Hill, 2011.

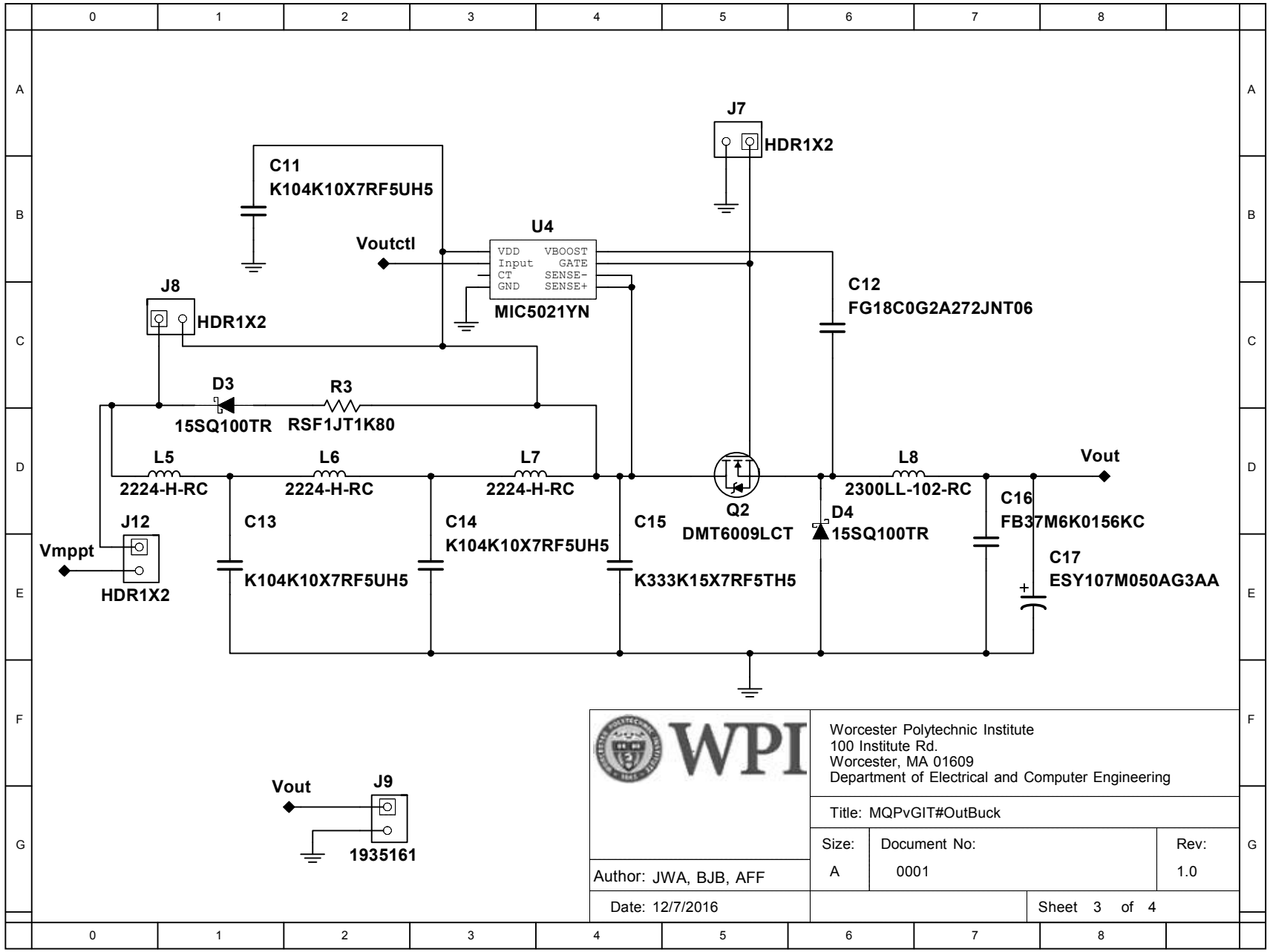
[13] W. Kim, V.-H. Duong, T.-T. Nguyen, and W. Choi, "Analysis of the effects of inverter ripple current on a photovoltaic power system by using an {AC} impedance model of the solar cell," *Renewable Energy*, vol. 59, pp. 150 – 157, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0960148113001912

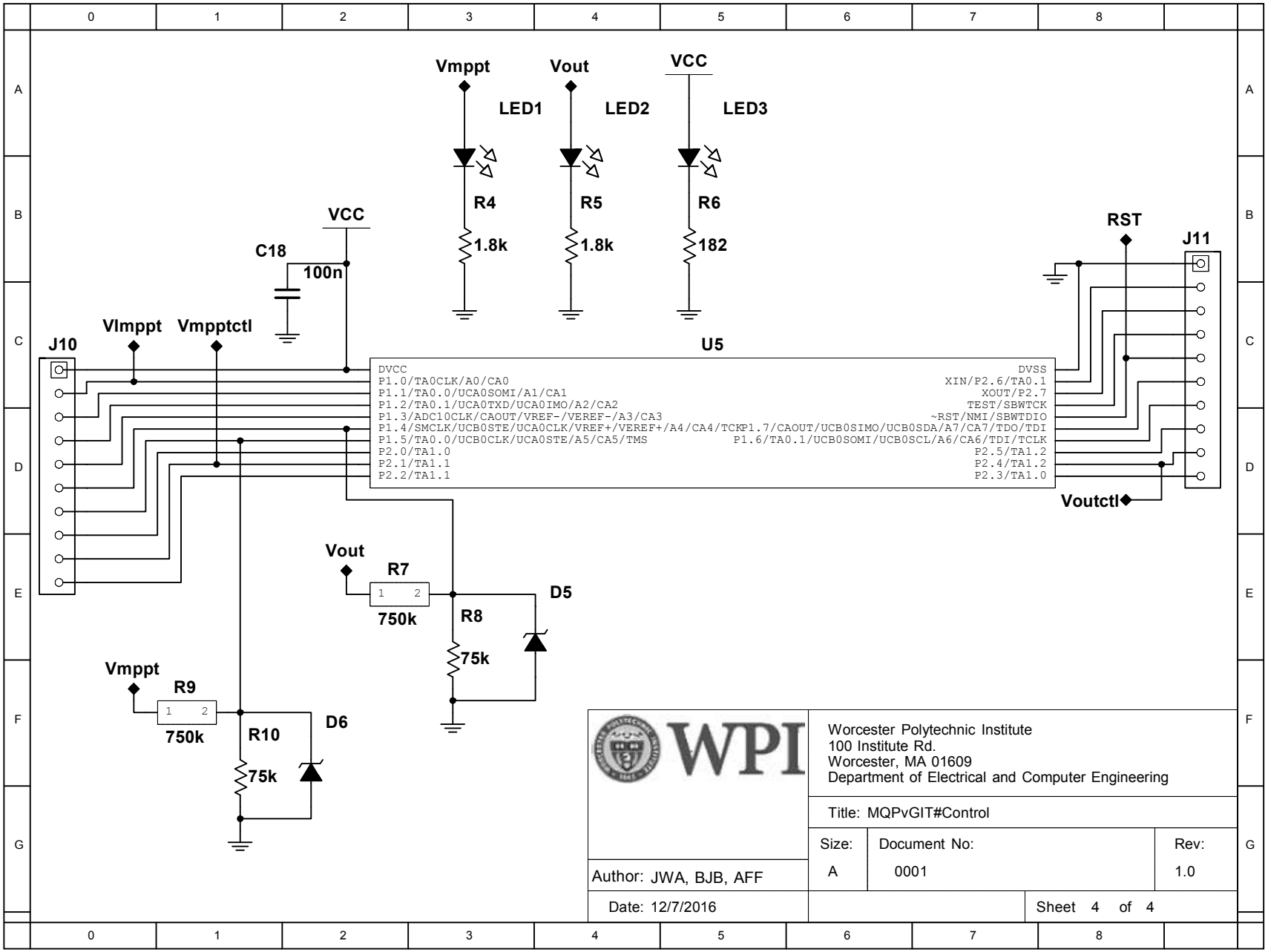
[14] "Msp430x2xx family user guide," 2013. [Online]. Available: http://www.ti.com/lit/ug/slau144j/slau144j.pdf

[15] J. Polonský, "Mightywatt kit: 70w electronic load for arduino by kaktus," 2015. [Online]. Available: https://www.tindie.com/products/Kaktus/mightywatt-kit-70w-electronic-load-for-arduino/


VCC
3.3V
J2
C2
K104K10X7RF5UH5
HDR1X2
Vmppt
J5
J4
1935161
HDR1X2
VCC
3.3V
C3
K104K10X7RF5UH5
R1
RNF14FTD1K00
RST
VCC
3.3V
C4
K104K10X7RF5UH5
Q1
DMT6009LCT
Vfilter+
L1
2300LL-102-RC
U1
VCC IP-
IP+ VIOUT
GND BW_SEL
ACS722LLCTR-10AU-T
Vmppt
C5
K104K10X7RF5UH5
C6
FG18C0G2A272JNT06
J3
HDR1X2
D1
15SQ100TR
C1
FB37M6K0156KC
C7
ESY107M050AG3AA
VCC
3.3V
U6B
VImppt
LT1632CN8
U2
VDD VBOOST
Input GATE
CT SENSE-
GND SENSE+
MIC5021YN
Vfilter-
R11
RNF14FTD10K0
R12
RNF14FTD20K0
Vmpptctl
Vfilter+
WPI
Worcester Polytechnic Institute
100 Institute Rd.
Worcester, MA 01609
Department of Electrical and Computer Engineering
Title: MQPvGIT#MPPTBuck
Size: A
Document No: 0001
Rev: 1.0
Author: JWA, BJB, AFF
Date: 12/7/2016
Sheet 1 of 4


D2
15SQ100TR
R2
RSF1JT1K80
L2
2224-H-RC
L3
2224-H-RC
L4
2224-H-RC
Vfilter+
J1
1935161
C9
K104K10X7RF5UH5
C8
K104K10X7RF5UH5
C10
K333K15X7RF5TH5
Vfilter-
VCC
3.3V
U3
L78L33ABZ-AP
Vout Vin
GND
J6
1
3
2
PJ-018
WPI
Worcester Polytechnic Institute
100 Institute Rd.
Worcester, MA 01609
Department of Electrical and Computer Engineering
Title: MQPvGIT#InputFilter
Size: Custom
Document No: 0001
Rev: 1.0
Author: JWA, BJB, AFF
Date: 12/7/2016
Sheet 2 of 4


J7
HDR1X2
C11
K104K10X7RF5UH5
U4
Voutctl
VDD VBOOST
Input GATE
CT SENSE-
GND SENSE+
MIC5021YN
J8
HDR1X2
C12
FG18C0G2A272JNT06
D3
15SQ100TR
R3
RSF1JT1K80
L5
2224-H-RC
L6
2224-H-RC
L7
2224-H-RC
L8
2300LL-102-RC
Vout
Q2
DMT6009LCT
D4
15SQ100TR
C16
FB37M6K0156KC
J12
Vmppt
HDR1X2
C13
K104K10X7RF5UH5
C14
K104K10X7RF5UH5
C15
K333K15X7RF5TH5
C17
ESY107M050AG3AA
Vout
J9
1935161
WPI
Worcester Polytechnic Institute
100 Institute Rd.
Worcester, MA 01609
Department of Electrical and Computer Engineering
Title: MQPvGIT#OutBuck
Size: A
Document No: 0001
Rev: 1.0
Author: JWA, BJB, AFF
Date: 12/7/2016
Sheet 3 of 4


Vmppt
Vout
VCC
LED1
LED2
LED3
R4
R5
R6
1.8k
1.8k
182
VCC
C18
100n
RST
J11
J10
VImppt
Vmpptctl
U5
DVCC
P1.0/TA0CLK/A0/CA0
P1.1/TA0.0/UCA0SOMI/A1/CA1
P1.2/TA0.1/UCA0TXD/UCA0IMO/A2/CA2
P1.3/ADC10CLK/CAOUT/VREF-/VEREF-/A3/CA3
P1.4/SMCLK/UCB0STE/UCA0CLK/VREF+/VEREF+/A4/CA4/TCKP1.7/CAOUT/UCB0SIMO/UCB0SDA/A7/CA7/TDO/TDI
P1.5/TA0.0/UCB0CLK/UCA0STE/A5/CA5/TMS
P2.0/TA1.0
P2.1/TA1.1
P2.2/TA1.1
DVSS
XIN/P2.6/TA0.1
XOUT/P2.7
TEST/SBWTCK
~RST/NMI/SBWTDIO
P1.6/TA0.1/UCB0SOMI/UCB0SCL/A6/CA6/TDI/TCLK
P2.5/TA1.2
P2.4/TA1.2
P2.3/TA1.0
Voutctl
Vout
R7
750k
R8
75k
D5
Vmppt
R9
750k
R10
75k
D6
WPI
Worcester Polytechnic Institute
100 Institute Rd.
Worcester, MA 01609
Department of Electrical and Computer Engineering
Title: MQPvGIT#Control
Size: A
Document No: 0001
Rev: 1.0
Author: JWA, BJB, AFF
Date: 12/7/2016
Sheet 4 of 4

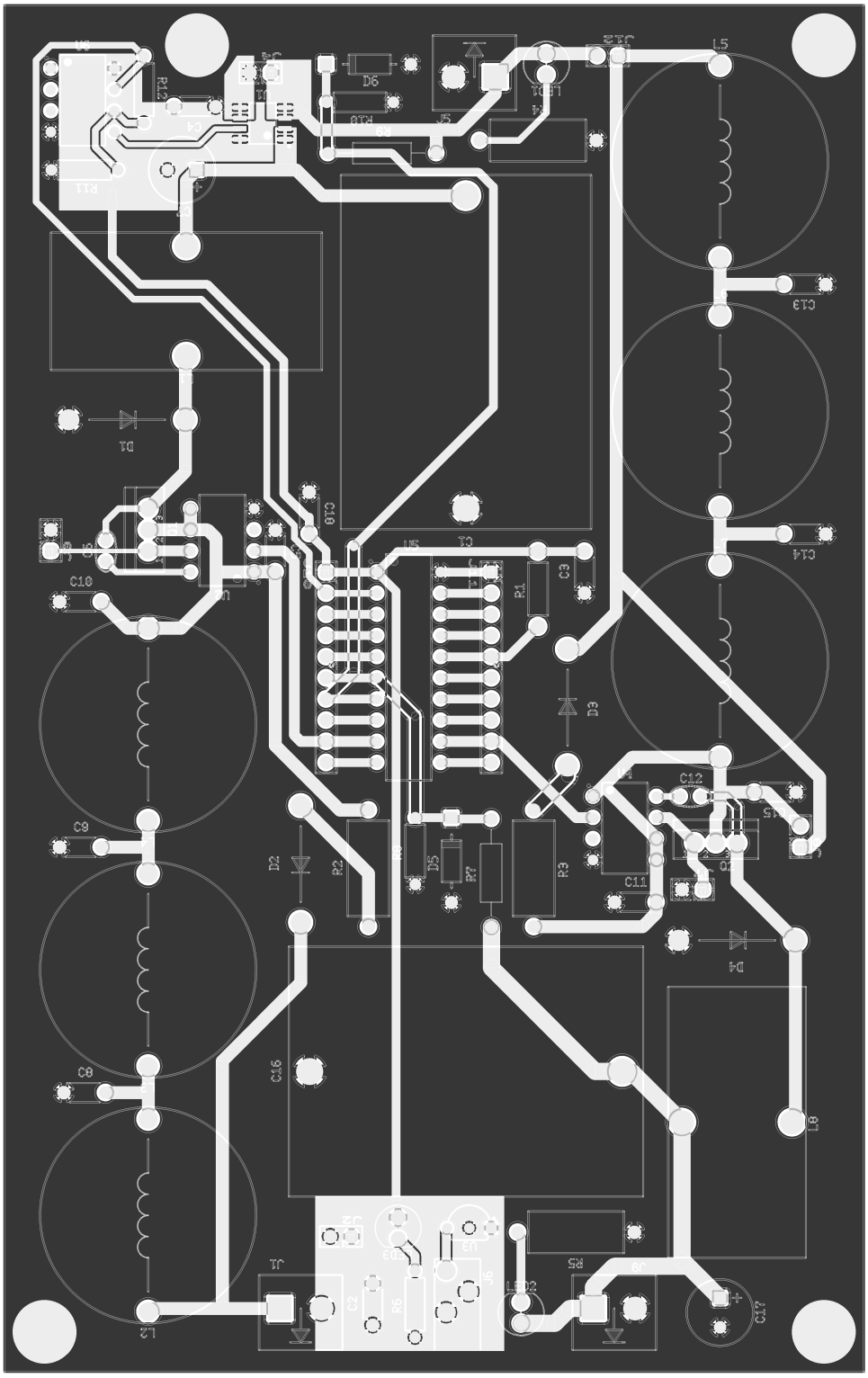
B Bill Of Materials

	Solar Maximum Power Point Tracker MQP BoM (Updated 1/20/17)				
	Worcester Polytechnic Institue				
	Authors: Johnathan Adams, Ben Beauregard, Andrew Flynn				
	Advisor: Prof. Alexander Emanuel				
Ref.	**Description**	**Value**	**MPN**	**Vendor**	**Vendor Part Num.**
R1	MCU resest pullup Resistor	1k Ohm	RNF14FTD1K00	Digi-Key	RNF14FTD1K00CT-ND
R2	MPPT Input Filter Freewheeling Current Limit Resistor	1.8k Ohm	RSF1JT1K80	Digi-Key	RSF1JT1K80CT-ND
R3	Outbuck Input Filter Freewheeling Current Limit Resistor	1.8k Ohm	RSF1JT1K80	Digi-Key	RSF1JT1K80CT-ND
R4	VMPPT LED current limit Resistor	1.8k Ohm	RSF1JT1K80	Digi-Key	RSF1JT1K80CT-ND
R5	VOUT LED current limit Resistor	1.8k Ohm	RSF1JT1K80	Digi-Key	RSF1JT1K80CT-ND
R6	VCC LED current limit Resistor	182 Ohm	RNF14FTD182R	Digi-Key	RNF14FTD182RCT-ND
R7	VOUT Voltage Sense divider (high side) Resistor	750k Ohm	RNF14FTD750K	Digi-Key	RNF14FTD750KCT-ND
R8	VOUT Voltage Sense divider (low side) Resistor	75k Ohm	RNF14FTD75K0	Digi-Key	RNF14FTD75K0CT-ND
R9	VMPPT Voltage Sense divider (high side) Resistor	750k Ohm	RNF14FTD750K	Digi-Key	RNF14FTD750KCT-ND
R10	VMPPT Voltage Sense divider (low side) Resistor	75k Ohm	RNF14FTD75K0	Digi-Key	RNF14FTD75K0CT-ND
R11	VIMPPT comp divider (low side) Resistor	10k Ohm	RNF14FTD10K0	Digi-Key	RNF14FTD10K0CT-ND
R12	VIMPPT comp divider (high side) Resistor	20k Ohm	RNF14FTD20K0	Digi-Key	RNF14FTD20K0CT-ND
C1	MPPT Output Cap 1	15uF, 1kV	FB37M6K0156KC	Digi-Key	478-9661-ND
C2	VCC Filter Cap	0.1uF, 50V	K104K10X7RF5UH5	Digi-Key	BC2665CT-ND
C3	VCC Filter Cap	0.1uF, 50V	K104K10X7RF5UH5	Digi-Key	BC2665CT-ND
C4	VCC Filter Cap	0.1uF, 50V	K104K10X7RF5UH5	Digi-Key	BC2665CT-ND
C5	MPPT MOSFET Driver VDD Filter Cap	0.1uF, 50V	K104K10X7RF5UH5	Digi-Key	BC2665CT-ND
C6	MPPT MOSFET Driver Boost Cap	2700pF, 100V	FG18C0G2A272JNT06	Digi-Key	445-173234-1-ND
C7	MPPT Output Cap 2	100uF, 50V	ESY107M050AG3AA	Digi-Key	ESY107M050AG3AA
C8	MPPT Input Filter Stage 1 Cap	0.1uF, 50V	K104K10X7RF5UH5	Digi-Key	BC2665CT-ND
C9	MPPT Input Filter Stage 2 Cap	0.1uF, 50V	K104K10X7RF5UH5	Digi-Key	BC2665CT-ND
C10	MPPT Input Filter Stage 3 Cap	0.033uF, 50V	K333K15X7RF5TH5	Digi-Key	BC1098CT-ND
C11	OutBuck MOSFET Driver VDD Filter Cap	0.1uF, 50V	K104K10X7RF5UH5	Digi-Key	BC2665CT-ND
C12	MOSFET Driver Boost Cap	2700pF, 100V	FG18C0G2A272JNT06	Digi-Key	445-173234-1-ND
C13	OutBuck Input Filter Stage 1 Cap	0.1uF, 50V	K104K10X7RF5UH5	Digi-Key	BC2665CT-ND
C14	OutBuck Input Filter Stage 2 Cap	0.1uF, 50V	K104K10X7RF5UH5	Digi-Key	BC2665CT-ND
C15	OutBuck Input Filter Stage 3 Cap	0.033uF, 50V	K333K15X7RF5TH5	Digi-Key	BC1098CT-ND
C16	OutBuck Output Cap 1	15uF, 1kV	FB37M6K0156KC	Digi-Key	478-9661-ND
C17	OutBuck Output Cap 2	100uF, 50V	ESY107M050AG3AA	Digi-Key	ESY107M050AG3AA
C18	VCC Filter	0.1uF, 50V	K104K10X7RF5UH5	Digi-Key	BC2665CT-ND
C19	Current Sense Amp Oscillation Compensation	10pF, 50V	K100J15C0GF5TL2	Digi-Key	BC1001CT-ND
L1	MPPT Output Inductor	1mH, 3.5A	2300LL-102-V-RC	Digi-Key	2300LL-102-V-RC-ND
L2	MPPT Input Filter Stage 1 Inductor	1mH, 1.9A	2224-H-RC	Digi-Key	M8830-ND
L3	MPPT Input Filter Stage 2 Inductor	1mH, 1.9A	2224-H-RC	Digi-Key	M8830-ND
L4	MPPT Input Filter Stage 3 Inductor	1mH, 1.9A	2224-H-RC	Digi-Key	M8830-ND
L5	OutBuck Input Filter Stage 1 Inductor	1mH, 1.9A	2224-H-RC	Digi-Key	M8830-ND
L6	OutBuck Input Filter Stage 2 Inductor	1mH, 1.9A	2224-H-RC	Digi-Key	M8830-ND
L7	OutBuck Input Filter Stage 3 Inductor	1mH, 1.9A	2224-H-RC	Digi-Key	M8830-ND
L8	OutBuck Output Inductor	1mH, 3.5A	2300LL-102-V-RC	Digi-Key	2300LL-102-V-RC-ND
D1	MPPT Output Diode	Shottky, 100V, 15A	15SQ100TR	Digi-Key	1655-1355-1-ND
D2	MPPT Input Filter Freewheeling Diode	Shottky, 100V, 15A	15SQ100TR	Digi-Key	1655-1355-1-ND
D3	OutBuck Input Filter Freewheeling Diode	Shottky, 100V, 15A	15SQ100TR	Digi-Key	1655-1355-1-ND
D4	OutBuck Output Diode	Shottky, 100V, 15A	15SQ100TR	Digi-Key	1655-1355-1-ND
D5	VOUT Clamping Diode	Zener, 3.6V, 500mW	BZX79C3V6	Digi-Key	BZX79C3V6-ND
D6	VMPPT Clamping Diode	Zener, 3.6V, 500mW	BZX79C3V6	Digi-Key	BZX79C3V6-ND
Q1	MPPT MOSFET	60V, 30A NMOS	DMT6009LCT	Digi-Key	DMT6009LCTDI-5-ND
Q2	OutBuck MOSFET	60V, 30A NMOS	DMT6009LCT	Digi-Key	DMT6009LCTDI-5-ND

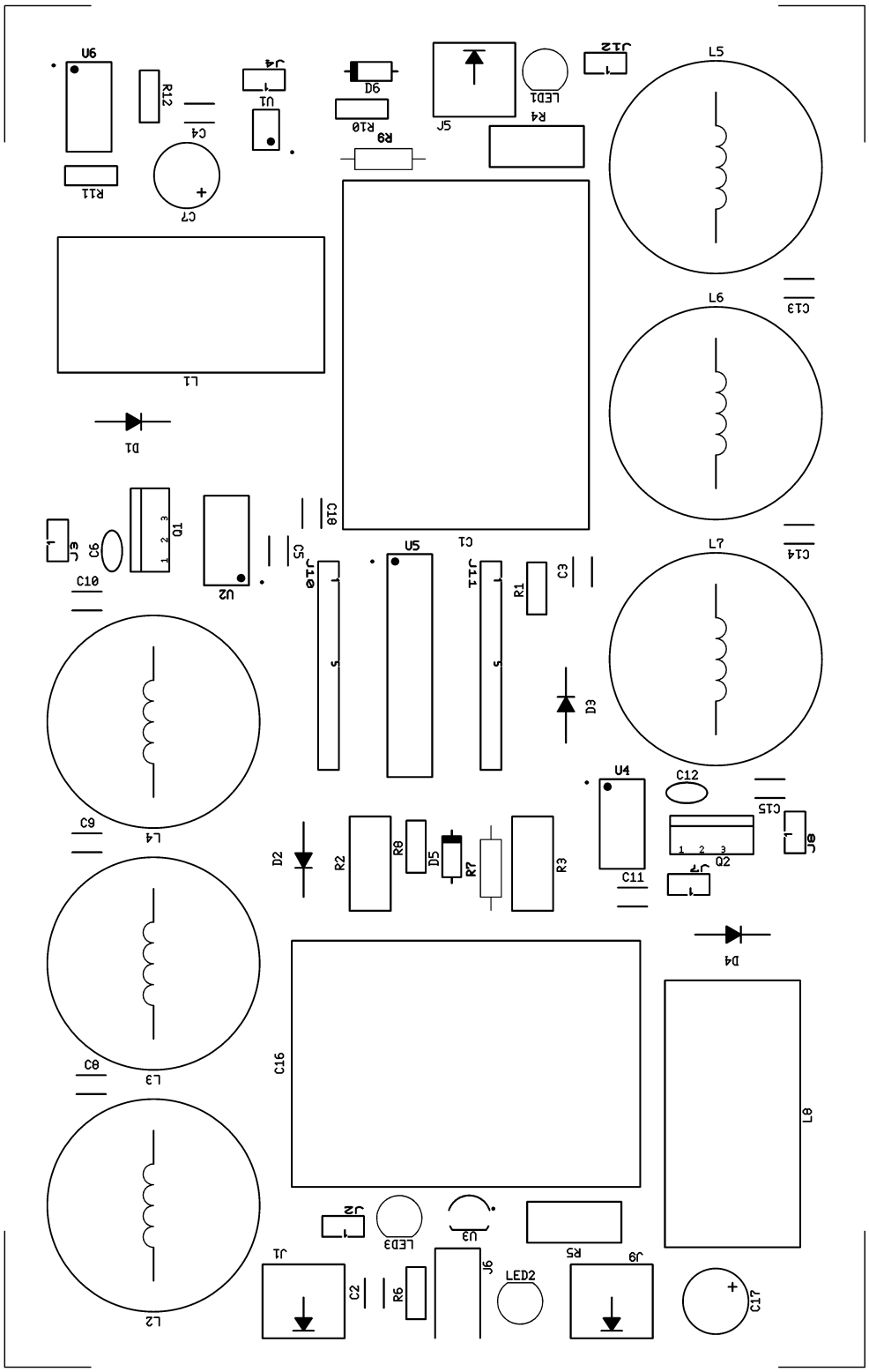
U1	MPPT Current Sensor		ACS722LLCTR-10AU-T	Digi-Key	620-1636-6-ND
U2	MPPT High Side MOSFET Driver		MIC5021YN	Digi-Key	576-1240-ND
U3	3.3V Linear Regulator		L78L33ABZ-AP	Digi-Key	497-7287-1-ND
U4	OutBuck High Side MOSFET Driver		MIC5021YN	Digi-Key	576-1240-ND
U5	Main MCU		MSP430G2553IN20	Digi-Key	296-28429-5-ND
U6	VIMPPT Comparator		LT1632CN8#PBF	Digi-Key	LT1632CN8#PBF-ND
J1	Solar Input Connector (to MPPT Input Filter)	2 pos. term block	1935161	Digi-Key	277-1667-ND
J2	VCC Connection Port (3.3V direct)	2 pos female header	M20-7820246	Digi-Key	952-1776-ND
J3	MPPT PWM Test Port	2 pos female header	M20-7820246	Digi-Key	952-1776-ND
J4	VMPPT (output of MPPT buck converter)	2 pos female header	M20-7820246	Digi-Key	952-1776-ND
J5	VMPPT (output of MPPT buck converter)	2 pos. term block	1935161	Digi-Key	277-1667-ND
J6	Aux VCC Input (to U3, 3.3V regulator)	1.7mm x 4mm jack	PJ-018	Digi-Key	PJ-018
J6-P	Barrel jack for Aux VCC Input	1.7mm x 4mm Fem.	PP-013	Digi-Key	CP-013-ND
J7	OutBuck PWM Test Port	2 pos female header	M20-7820246	Digi-Key	952-1776-ND
J8	OutBuck Input Filter Freehwleeling Diode Bypass Jumper	2 pos male header	M20-9990246	Digi-Key	952-2262-ND
J9	VOUT (output of OutBuck buck converter)	2 pos. term block	1935161	Digi-Key	277-1667-ND
J10	MCU Breakout	10 pos female header	M20-7821046	Digi-Key	952-1847-ND
J11	MCU Breakout	10 pos female header	M20-7821046	Digi-Key	952-1847-ND
J12	VMPPT to OutBuck Input Filter Jumper	2 pos male header	M20-9990246	Digi-Key	952-2262-ND
SH1	Shunt for J8 (diode bypass)	Blue 2 pos shunt	M7683-05	Digi-Key	952-2572-ND
SH2	Shunt for J12 (VMPPT to OutBuck)	Red 2 pos shunt	M7681-05	Digi-Key	952-1855-ND
LED1	VMPPT Indicator	Green 3mm, 2.2V	151031VS06000	Digi-Key	732-5008-ND
LED2	VOUT Indicator	Green 3mm, 2.2V	151031VS06000	Digi-Key	732-5008-ND
LED3	VCC Incidator	Green 3mm, 2.2V	151031VS06000	Digi-Key	732-5008-ND
SK1	IC Socket for U5 (MCU)	2 x 10 pins	SA203000	Digi-Key	ED3020-ND
PCB	Custom-designed PCB	2-layer		Advanced Circuits	

C Gerber (RS-274x) File Renders

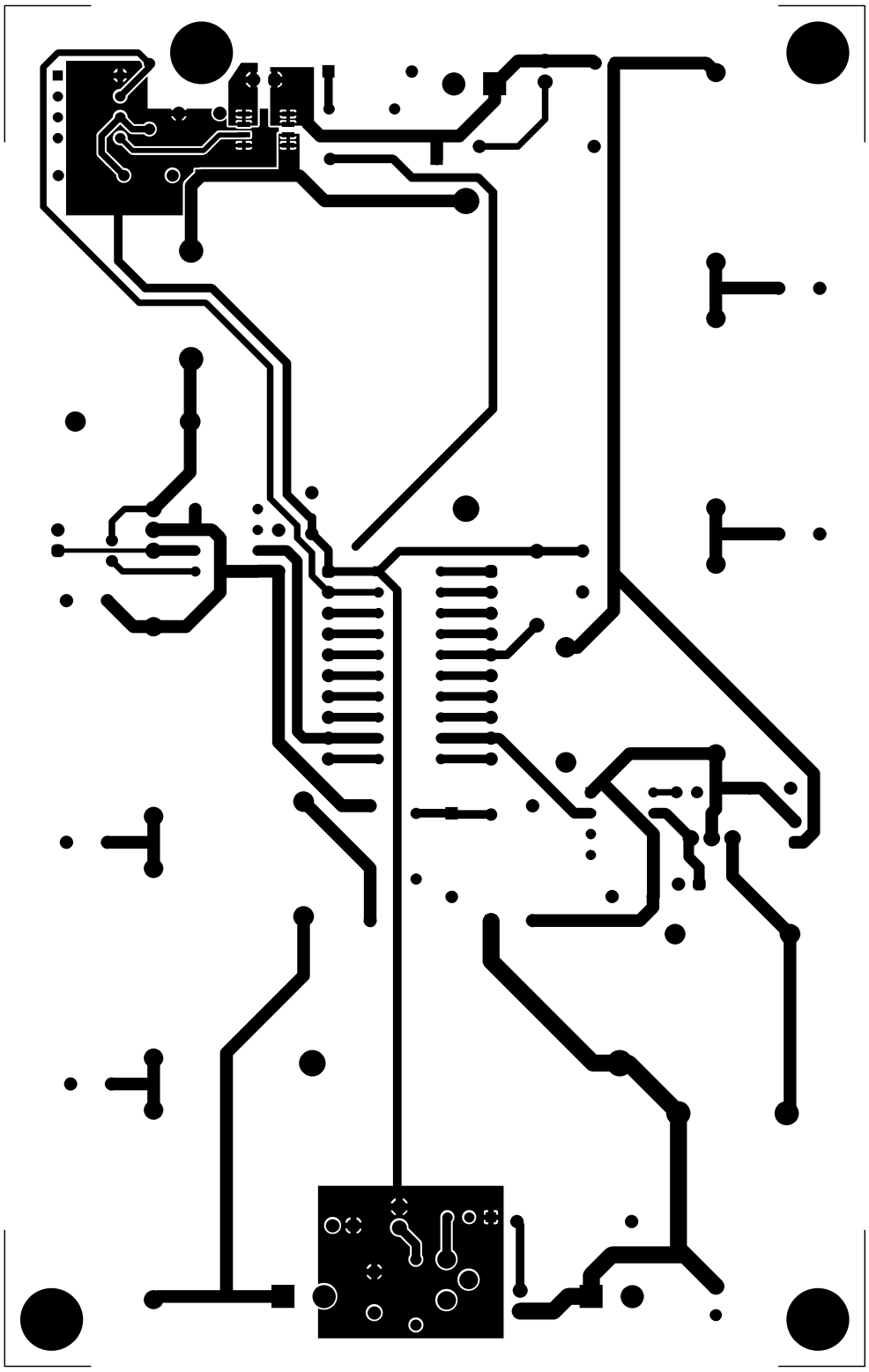
Images of gerber files generated by Online Gerber Viewer, http://www.gerber-viewer.com/.



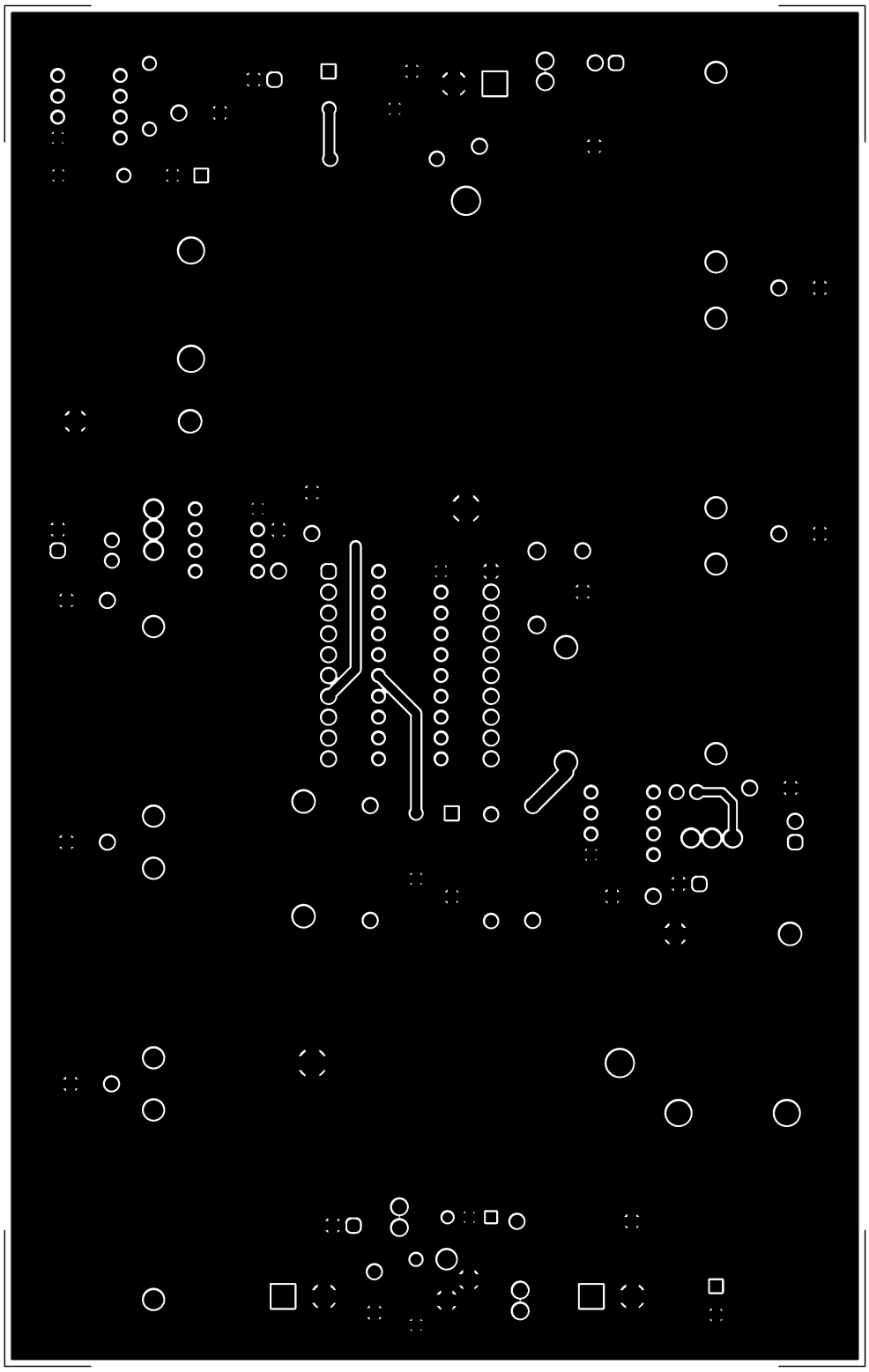
Top Silkscreen Gerber File Render



Top Copper Gerber File Render



Bottom Copper Gerber File Render



D PSpice Code

12/15/16 06:49:05 \\storage.wpi.edu\home\My_Documents\MPPT_GIT\MPPT\PSpice\FULL_CIR_SIM.CIR

```
FULL_CIR_SIM
**********************************************************************
* SOLAR MAX. POWER POINT TRACKER MQP
* JOHNATHAN ADAMS, BENJAMIN BEAUREGARD, ANDREW FLYNN
* ADVISOR: PROFESSOR ALEXANDER EMANUEL
* A, B, C TERM 2016-2017
* WORCESTER POLYTECHNIC INSTITUTE
**********************************************************************
* DOCUMENT: FULL_CIR_SIM.CIR
* DATE CREATED: 10/12/16

**********************************************************************
* INCLUDE: LIBRARY AND EXTERNAL FILE INCLUDES
**********************************************************************
.LIB MPPT_MODELS.LIB
.LIB PWM_STEP_SOURCE_320.LIB
.LIB PWM_STEP_SOURCE_320_PERCENT.LIB
.LIB NOTE.LIB

**********************************************************************
* MAIN: FOR CONNECTING PRIMARY SUBCIRCUITS TOGETHER
**********************************************************************
***SOLAR CELL EMULATOR***
*XSOLAR 1 0 SOLAR_SIM

***OPTIONAL TEST VOLTAGE IN PLACE OF SOLAR CELL EMULATOR***
VTEST 1 0 30

***FIRST FILTER***
XFILTER_1 1 2 0 FILTER_1

***MPPT BUCK CONVERTER AND STEPPED PWM DRIVER***
*USE XV1 FOR STEPPED PWM, VPWM FOR STATIC
XMPPT_BUCK 2 3 4 5 0 MPPT_BUCK
*XV1 3 4 PWM
VPWM 3 4 PULSE(0 3.3 0 1n 1n 0.015m 0.02m)
CPWM1 3 0 10n

***VOLTAGE REPRESENTING PERCENT DUTY CYCLE OF PWM SOURCE***
XVPERCENT 100 0 PERCENT

***OPTIONAL TEST LOAD FOR MPPT BUCK CONVERTER***
*RMPPT 5 0 50

***SECOND FILTER***
XFILTER_2 5 6 0 FILTER_2

***VOLTAGE REGULATOR BUCK CONVERTER AND STATIC PWM DRIVER***
XOUT_BUCK 6 7 8 9 0 OUT_BUCK
VPWM2 7 8 PULSE(0 3.3 0 1n 1n 0.015m 0.02m)
CPWM2 7 0 10n

***TEST LOAD FOR VOLTAGE REGULATOR BUCK CONVERTER***
RLOAD 9 0 10

```

```
56  ********************************************************************************
57  * SOLAR CELL EMULATOR
58  * OUTPUT NODE TO BE CONNECTED TO FIRST FILTER INPUT
59  * GND TO BE CONNECTED TO NODE 0
60  * R1 AND R2 MAKE UP 25% 10K POT TO IRF9520 GATE
61  ********************************************************************************
62  .SUBCKT SOLAR_SIM P_OUT GND
63      VSOLAR 1 GND 30
64      D1 1 2 D1N4004
65      R1 2 3 7.5K
66      R2 3 4 2.5K
67      R3 4 GND 10K
68      XQ1 P_OUT 3 2 IRF9520
69  .ENDS SOLAR_SIM
70
71  ********************************************************************************
72  * FILTER 1
73  * INPUT NODE TO BE CONNECTED TO SOLAR CELL EMULATIOR OUTPUT
74  * OUTPUT NODE TO BE CONNECTED TO VREG BUCK CONVERTER INPUT
75  ********************************************************************************
76  .SUBCKT FILTER_1 P_IN P_OUT GND
77      XFLTR1 P_IN 1 GND FLTR_STAGE
78      XFLTR2 1 2 GND FLTR_STAGE
79      L_FINAL 2 3 1m
80      RL_FINAL 3 P_OUT 10m
81      C_FINAL 4 GND 33n
82      RC_FINAL P_OUT 4 8m
83      D_FREE 5 P_IN MBR1060
84      RD P_OUT 5 100
85  .ENDS FILTER_1
86
87  .SUBCKT FLTR_STAGE 1 2 3
88      * External node designations:
89      * Node 1 -> stage input
90      * Node 2 -> stage output
91      * Node 3 -> ground
92      L1 1 4 1m
93      RL1 4 2 10m
94      C1 5 3 100n
95      RC1 5 2 8m
96  .ENDS FLTR_STAGE
97
98  ********************************************************************************
99  * MPPT BUCK CONVERTER
100 * INPUT NODE TO BE CONNECTED TO FILTER 1 OUTPUT
101 * PWM SOURCE IN MAIN GETS CONNECTED BETWEEN SIG_IN AND P_REF
102 * OUTPUT NODE TO BE CONNECTED TO FILTER 2
103 ********************************************************************************
104 .SUBCKT MPPT_BUCK P_IN SIG_IN P_REF P_OUT GND
105     XQ2 P_IN SIG_IN P_REF DMT6009LCT
106     D2 GND P_REF MBR1060
107     L1 P_REF 1 1m
108     RL1 1 P_OUT 137m
109     C1 2 GND 15u
110     RC1 P_OUT 2 8m
111     C2 3 GND 100u
112     RC2 P_OUT 3 74m
```

```
.ENDS MPPT_BUCK

*********************************************************************
* FILTER 2
* INPUT NODE TO BE CONNECTED TO MPPT BUCK CONVERTER OUTPUT
* OUTPUT NODE TO BE CONNECTED TO OUTPUT BUCK CONVERTER INPUT
* NOTE: USES "FLTR_STAGE" SUBCIRCUIT FROM FILTER 1
*********************************************************************
.SUBCKT FILTER_2 P_IN P_OUT GND
    XFLTR1 P_IN 1 GND FLTR_STAGE
    XFLTR2 1 2 GND FLTR_STAGE
    L_FINAL 2 3 1m
    RL_FINAL 3 P_OUT 10m
    C_FINAL 4 GND 33n
    RC_FINAL P_OUT 4 8m
    D_FREE 5 P_IN MBR1060
    RD P_OUT 5 100
.ENDS FILTER_2

*********************************************************************
* OUT BUCK CONVERTER
* P_IN AND GND NODES TO BE CONNECTED TO FILTER_2 OUTPUT
* SIG_IN NODE TO BE CONNECTED TO PULSE VOLTAGE SOURCE IN MAIN FOR PWM
* OUTPUT NODES TO BE CONNECTED TO LOAD (IN MAIN)
* NOTE: USES DMT6009LCT SUBCIRCUIT FROM MPPT_BUCK
*********************************************************************
.SUBCKT OUT_BUCK P_IN SIG_IN P_REF P_OUT GND
    XQ3 P_IN SIG_IN P_REF DMT6009LCT
    D3 GND P_REF MBR1060
    L1 P_REF 1 1m
    RL1 1 P_OUT 137m
    C1 2 GND 15u
    RC1 P_OUT 2 8m
    C2 3 GND 100u
    RC2 P_OUT 3 74m
.ENDS OUT_BUCK

*********************************************************************
* SIMULATION CONFIGURATION
*********************************************************************
.PROBE
.TRAN 1580m 1580m 0u 1u UIC
.END
```

Custom PSpice Device Models Library: MPPT_MODELS.LIB

12/14/16 02:17:29 \\storage.wpi.edu\home\My_Documents\MPPT_GIT\MPPT\PSpice\MPPT_MODELS.LIB

```
1  ********************************************************************************
2  * SOLAR MAX. POWER POINT TRACKER MQP
3  * JOHNATHAN ADAMS, BENJAMIN BEAUREGARD, ANDREW FLYNN
4  * ADVISOR: PROFESSOR ALEXANDER EMANUEL
5  * A, B, C TERM 2016-2017
6  * WORCESTER POLYTECHNIC INSTITUTE
7  ********************************************************************************
8  * DOCUMENT: MPPT_MODELS.LIB
9  * DATE CREATED: 11/4/16
10
11 * MODEL BY SYMMETRY DESIGN SYSTEMS
12 .MODEL D1N4004 D
13     +IS=5.31656e-08 RS=0.0392384 N=2 EG=0.6
14     +XTI=0.05 BV=400 IBV=5e-08 CJO=1e-11
15     +VJ=0.7 M=0.5 FC=0.5 TT=1e-09
16     +KF=0 AF=1
17
18 * MODEL BY SYMMETRY DESIGN SYSTEMS
19 .MODEL MBR1060 D
20     +IS=0.000132385 RS=0.0122186 N=2 EG=0.828762
21     +XTI=3.80757 BV=60 IBV=0.0001 CJO=1e-11
22     +VJ=0.7 M=0.5 FC=0.5 TT=0
23     +KF=0 AF=1
24
25 .SUBCKT IRF9520 1 2 3
26     *MODEL BY SYMMETRY DESIGN SYSTEMS
27     * External Node Designations
28     * Node 1 -> Drain
29     * Node 2 -> Gate
30     * Node 3 -> Source
31     M1 9 7 8 8 MM L=100u W=100u
32     * Default values used in MM:
33     * The voltage-dependent capacitances are
34     * not included. Other default values are:
35     *   RS=0 RD=0 LD=0 CBD=0 CBS=0 CGBO=0
36     .MODEL MM PMOS LEVEL=1 IS=1e-32
37     +VTO=-3.41185 LAMBDA=0.0289226 KP=3.46967
38     +CGSO=3.45033e-06 CGDO=1e-11
39     RS 8 3 0.167957
40     D1 1 3 MD
41     .MODEL MD D IS=7.308e-22 RS=0.17 N=1.29916 BV=100
42     +IBV=10 EG=1 XTI=1 TT=1e-07
43     +CJO=4.53963e-10 VJ=2.47692 M=0.539653 FC=0.1
44     RDS 3 1 1e+06
45     RD 9 1 0.198401
46     RG 2 7 11.3744
47     D2 5 4 MD1
48     * Default values used in MD1:
49     *   RS=0 EG=1.11 XTI=3.0 TT=0
50     *   BV=infinite IBV=1mA
51     .MODEL MD1 D IS=1e-32 N=50
52     +CJO=3.45426e-10 VJ=1.57654 M=0.730307 FC=1e-08
53     D3 5 0 MD2
54     * Default values used in MD2:
55     *   EG=1.11 XTI=3.0 TT=0 CJO=0
```

```
56  ********************************************************************************
57  * SOLAR CELL EMULATOR
58  * OUTPUT NODE TO BE CONNECTED TO FIRST FILTER INPUT
59  * GND TO BE CONNECTED TO NODE 0
60  * R1 AND R2 MAKE UP 25% 10K POT TO IRF9520 GATE
61  ********************************************************************************
62  .SUBCKT SOLAR_SIM P_OUT GND
63      VSOLAR 1 GND 30
64      D1 1 2 D1N4004
65      R1 2 3 7.5K
66      R2 3 4 2.5K
67      R3 4 GND 10K
68      XQ1 P_OUT 3 2 IRF9520
69  .ENDS SOLAR_SIM
70
71  ********************************************************************************
72  * FILTER 1
73  * INPUT NODE TO BE CONNECTED TO SOLAR CELL EMULATIOR OUTPUT
74  * OUTPUT NODE TO BE CONNECTED TO VREG BUCK CONVERTER INPUT
75  ********************************************************************************
76  .SUBCKT FILTER_1 P_IN P_OUT GND
77      XFLTR1 P_IN 1 GND FLTR_STAGE
78      XFLTR2 1 2 GND FLTR_STAGE
79      L_FINAL 2 3 1m
80      RL_FINAL 3 P_OUT 10m
81      C_FINAL 4 GND 33n
82      RC_FINAL P_OUT 4 8m
83      D_FREE 5 P_IN MBR1060
84      RD P_OUT 5 100
85  .ENDS FILTER_1
86
87  .SUBCKT FLTR_STAGE 1 2 3
88      * External node designations:
89      * Node 1 -> stage input
90      * Node 2 -> stage output
91      * Node 3 -> ground
92      L1 1 4 1m
93      RL1 4 2 10m
94      C1 5 3 100n
95      RC1 5 2 8m
96  .ENDS FLTR_STAGE
97
98  ********************************************************************************
99  * MPPT BUCK CONVERTER
100 * INPUT NODE TO BE CONNECTED TO FILTER 1 OUTPUT
101 * PWM SOURCE IN MAIN GETS CONNECTED BETWEEN SIG_IN AND P_REF
102 * OUTPUT NODE TO BE CONNECTED TO FILTER 2
103 ********************************************************************************
104 .SUBCKT MPPT_BUCK P_IN SIG_IN P_REF P_OUT GND
105     XQ2 P_IN SIG_IN P_REF DMT6009LCT
106     D2 GND P_REF MBR1060
107     L1 P_REF 1 1m
108     RL1 1 P_OUT 137m
109     C1 2 GND 15u
110     RC1 P_OUT 2 8m
111     C2 3 GND 100u
112     RC2 P_OUT 3 74m
```

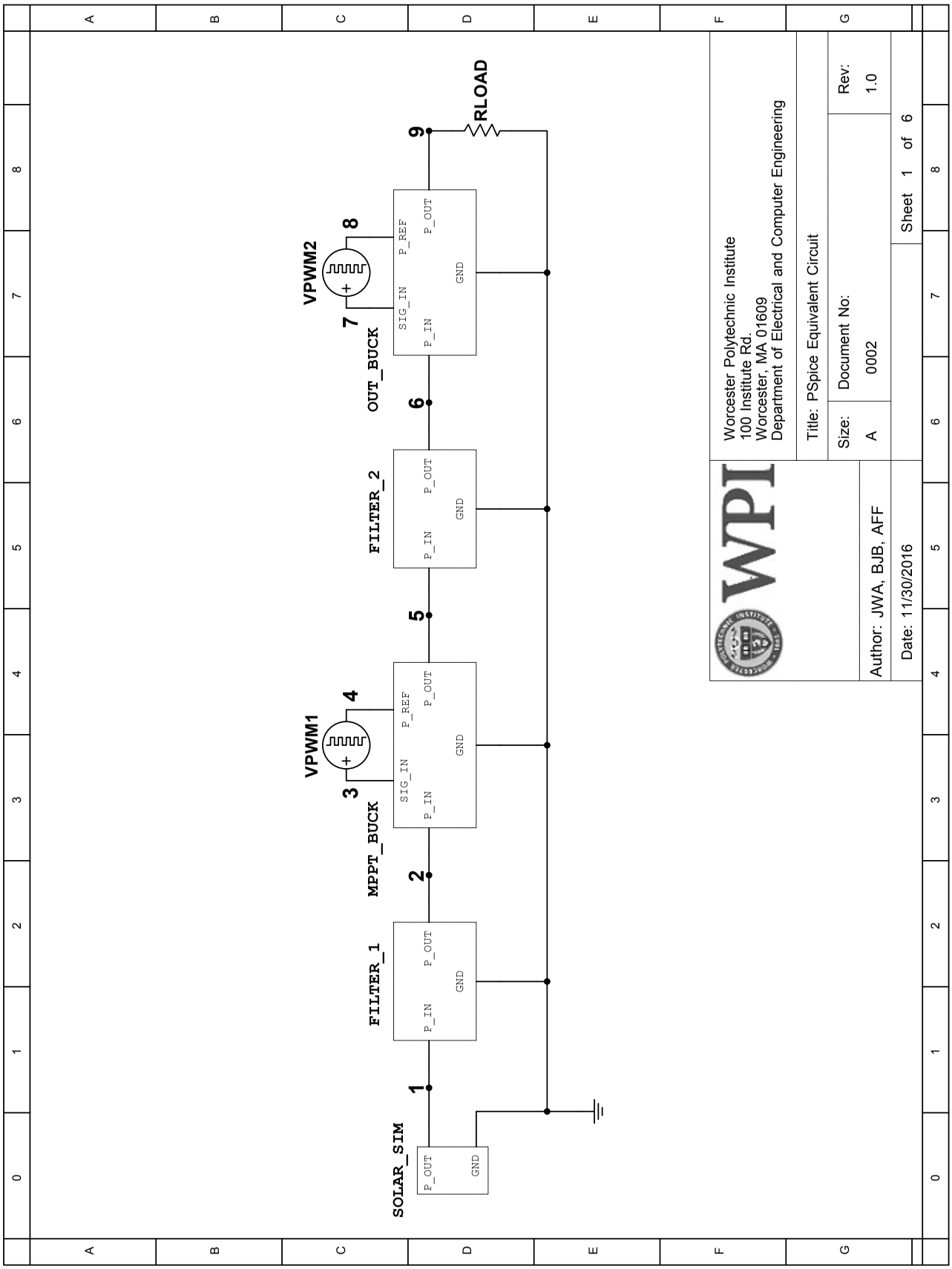
PWM Source Script: MPPT_PWM_LIB_GENERATOR.py

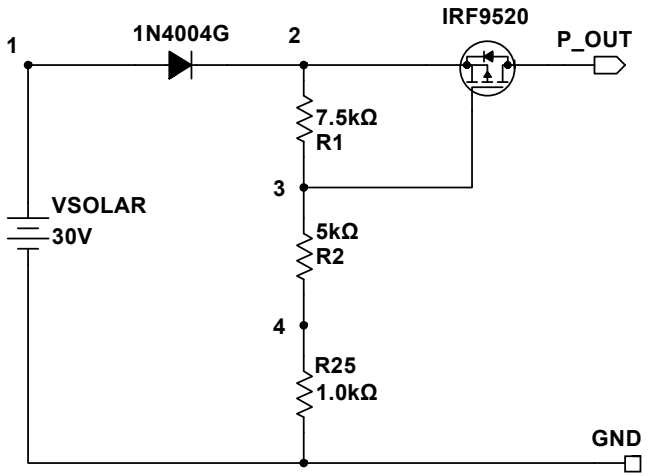
12/14/16 02:17:29 \\storage.wpi.edu\home\My_Documents\MPPT_GIT\MPPT\PSpice\MPPT_MODELS.LIB

```
 1  **************************************************************************
 2  * SOLAR MAX. POWER POINT TRACKER MQP
 3  * JOHNATHAN ADAMS, BENJAMIN BEAUREGARD, ANDREW FLYNN
 4  * ADVISOR: PROFESSOR ALEXANDER EMANUEL
 5  * A, B, C TERM 2016-2017
 6  * WORCESTER POLYTECHNIC INSTITUTE
 7  **************************************************************************
 8  * DOCUMENT: MPPT_MODELS.LIB
 9  * DATE CREATED: 11/4/16
10
11  * MODEL BY SYMMETRY DESIGN SYSTEMS
12  .MODEL D1N4004 D
13      +IS=5.31656e-08 RS=0.0392384 N=2 EG=0.6
14      +XTI=0.05 BV=400 IBV=5e-08 CJO=1e-11
15      +VJ=0.7 M=0.5 FC=0.5 TT=1e-09
16      +KF=0 AF=1
17
18  * MODEL BY SYMMETRY DESIGN SYSTEMS
19  .MODEL MBR1060 D
20      +IS=0.000132385 RS=0.0122186 N=2 EG=0.828762
21      +XTI=3.80757 BV=60 IBV=0.0001 CJO=1e-11
22      +VJ=0.7 M=0.5 FC=0.5 TT=0
23      +KF=0 AF=1
24
25  .SUBCKT IRF9520 1 2 3
26      *MODEL BY SYMMETRY DESIGN SYSTEMS
27      * External Node Designations
28      * Node 1 -> Drain
29      * Node 2 -> Gate
30      * Node 3 -> Source
31      M1 9 7 8 8 MM L=100u W=100u
32      * Default values used in MM:
33      * The voltage-dependent capacitances are
34      * not included. Other default values are:
35      *   RS=0 RD=0 LD=0 CBD=0 CBS=0 CGBO=0
36      .MODEL MM PMOS LEVEL=1 IS=1e-32
37      +VTO=-3.41185 LAMBDA=0.0289226 KP=3.46967
38      +CGSO=3.45033e-06 CGDO=1e-11
39      RS 8 3 0.167957
40      D1 1 3 MD
41      .MODEL MD D IS=7.308e-22 RS=0.17 N=1.29916 BV=100
42      +IBV=10 EG=1 XTI=1 TT=1e-07
43      +CJO=4.53963e-10 VJ=2.47692 M=0.539653 FC=0.1
44      RDS 3 1 1e+06
45      RD 9 1 0.198401
46      RG 2 7 11.3744
47      D2 5 4 MD1
48      * Default values used in MD1:
49      *   RS=0 EG=1.11 XTI=3.0 TT=0
50      *   BV=infinite IBV=1mA
51      .MODEL MD1 D IS=1e-32 N=50
52      +CJO=3.45426e-10 VJ=1.57654 M=0.730307 FC=1e-08
53      D3 5 0 MD2
54      * Default values used in MD2:
55      *   EG=1.11 XTI=3.0 TT=0 CJO=0
```

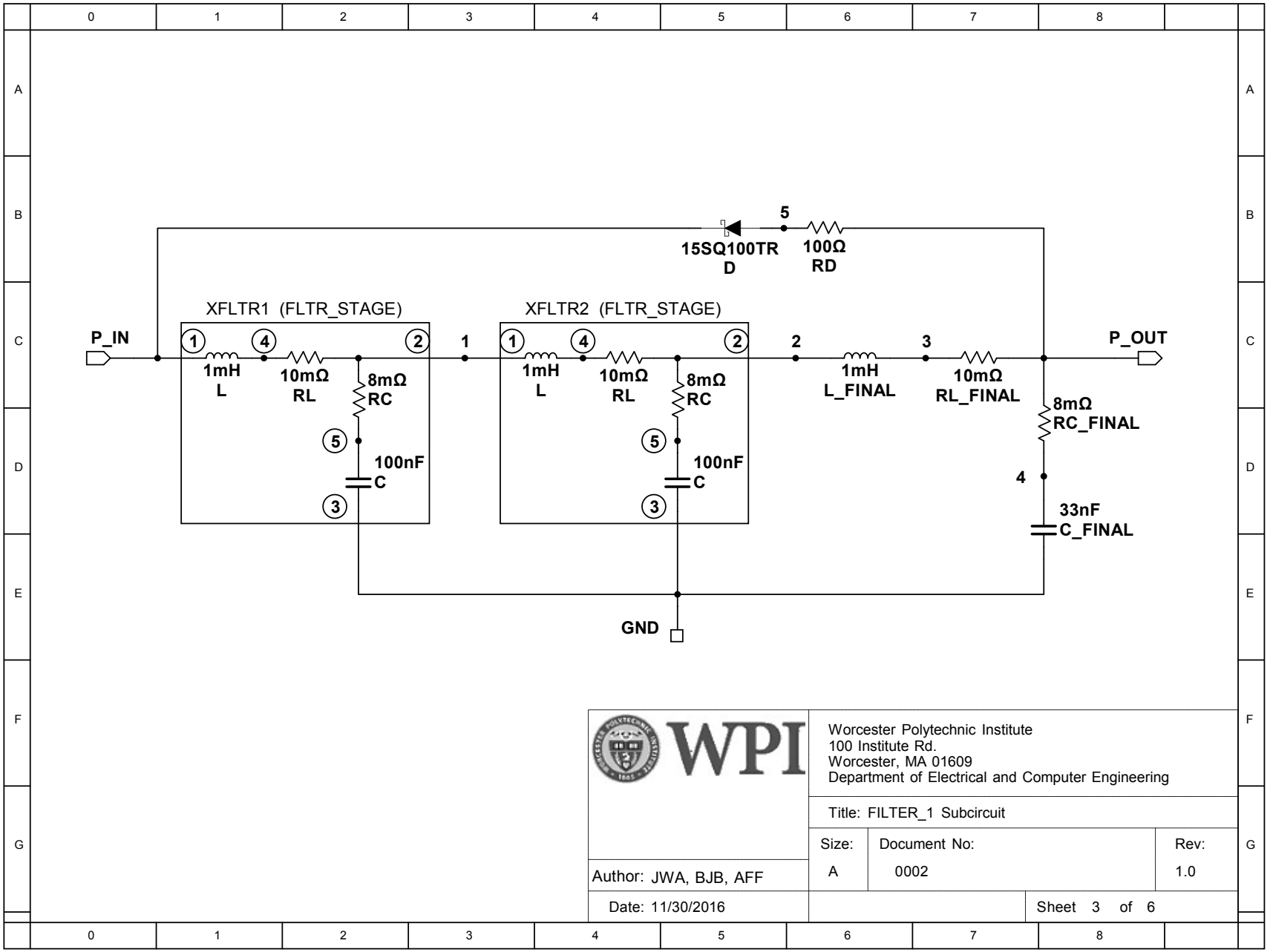
```
59      while (telap2-telap1 < tstep):
60          print(string.format(telap2,vhigh,telap2+DT-ttrans,vhigh), file=f)
61          print(string.format(telap2+DT,vlow,telap2+period-ttrans,vlow),
    file=f)
62          telap2 += period
63      pcurrent += pstep
64      DT = (pcurrent/100)*period
65      telap1=telap2
66
67  #finalize and close up pwm lib file
68  print(")", file=f)
69  print(".ENDS PWM", file=f)
70  f.close()
71
72  #finalize and close up pwm percent file
73  print(")", file=f2)
74  print(".ENDS PERCENT", file=f2)
75  f2.close()
```

E Equivalent PSpice Schematics

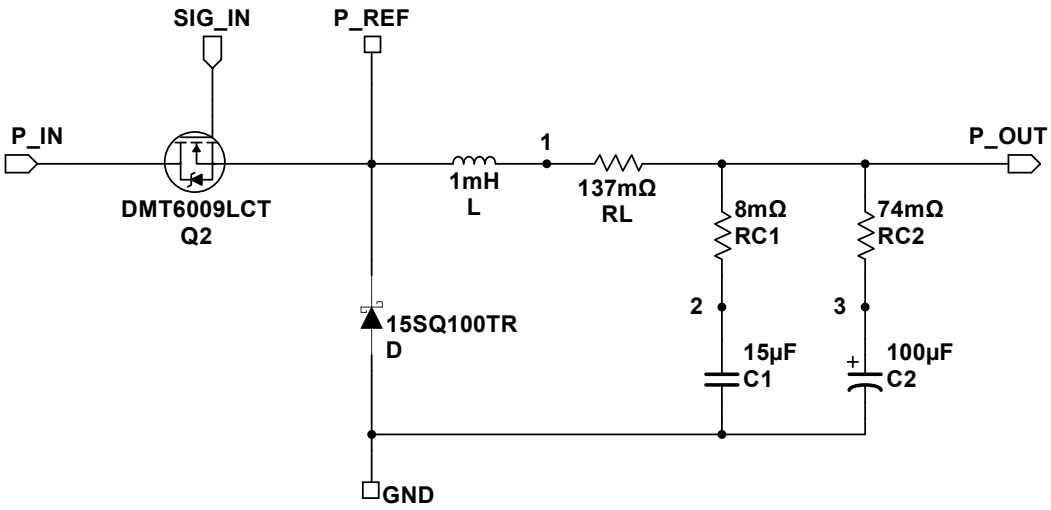




WPI	Worcester Polytechnic Institute 100 Institute Rd. Worcester, MA 01609 Department of Electrical and Computer Engineering		
	Title: SOLAR_SIM Subcircuit		
	Size: A	Document No: 0002	Rev: 1.0
Author: JWA, BJB, AFF			
Date: 11/30/2016		Sheet 2 of 6	



WPI	Worcester Polytechnic Institute 100 Institute Rd. Worcester, MA 01609 Department of Electrical and Computer Engineering		
	Title: FILTER_1 Subcircuit		
	Size: A	Document No: 0002	Rev: 1.0
Author: JWA, BJB, AFF			
Date: 11/30/2016		Sheet 3 of 6	



WPI	Worcester Polytechnic Institute 100 Institute Rd. Worcester, MA 01609 Department of Electrical and Computer Engineering	
	Title: MPPT_BUCK Subcircuit	
	Size: A	Document No: 0002 / Rev: 1.0
Author: JWA, BJB, AFF		
Date: 11/30/2016		Sheet 4 of 6

F C Source Code

```
1  /*******************************************************************************
2   *                    Maximum Power Point Tracker (MPPT)
3   *
4   * This code implements multiple algorithms for finding and tracking the
5   * maximum power point of a solar panel.
6   *
7   * This project is in fulfillment of the Major Qualifying Project at
8   * Worcester Polytechnic Institute.
9   *
10  * Authored by:
11  *         Johnathan Adams
12  *         Ben Beauregard
13  *         Andrew Flynn
14  *
15  *******************************************************************************/
16 // Includes
17 #include <msp430.h>
18 #include <limits.h>
19 #include "main.h"
20 #include "sweep.h"
21 //#include "beta.h"
22 #include "perturbobserve.h"
23
24 /* Define Global Variables */
25 // Store V-OUT from ADC
26 unsigned int v_out;
27 // Store V-OUT samples
28 int v_out_samples[AVERAGE_LENGTH];
29 // Saturation-Flag for integral computation (part of PID)
30 signed char v_out_sat;
31 // Value of Vout integral computation
32 long v_out_integral;
33 // Integration "parameter" (constant, multiplies error in integral calculation)
34 const int v_out_i = 1024;
35 // Store V-MPPT average from ADC
36 unsigned int v_mppt;
37 // Store V-MPPT samples
38 int v_mppt_samples[AVERAGE_LENGTH];
39 // Store I-MPPT average from ADC
40 unsigned int i_mppt;
41 // Store I-MPPT samples
42 int i_mppt_samples[AVERAGE_LENGTH];
43
44 // Proportional Constant = 1/2^Divisor
45 const int Divisor = 3;
```

```
// Counts the number of samples used for any average computation
unsigned int sample;
// Cycle counter for handling 0V input condition
unsigned char zero_samples;
// MPPT Duty Cycle - Start at 25%
int mppt_duty_cycle=80;

//Bitmask (see header).
volatile char DCTL;
// BITMASK of Buttons being pressed
volatile char BUTTONS;

// Calculated power draw from MPPT Buck Converter
unsigned long power;

// Define algorithm variable to choose MPPT algorithm
enum mppt_algorithm_type_enum algorithm = MPPT_SWEEP;

// State variable for MPPT state machine
enum mppt_states_enum mppt_state = MPPT_WAIT;
// State variable for VOUT state machine
enum vout_states_enum vout_state = VOUT_WAIT;

void main(void) {

    // Stop watchdog timer
    WDTCTL = WDTPW | WDTHOLD;
    /*
     * Configure Clocks
     */
    // 16MHz calibrated clock
    DCOCTL = CALDCO_16MHZ;
    // Set ACLK to /2
    BCSCTL1 = (CALBC1_16MHZ);
    // MCLK & SMCLK from DCO, both divided by 1, DCO resistor internal
    BCSCTL2 = 0b00000000;
    // Use VLOCLK for ACLK
    BCSCTL3 = LFXT1S_2;

    /*  PIN MAP
     * P1.0 - I - I-MPPT - A0
     * P1.1 - I - Algorithm Reset Button
     * P1.2 - I - Algorithm Change Button
     * P1.3 - O - UNUSED
     * P1.4 - I - V-OUT - A4
     * P1.5 - I - V-MPPT - A5
     * P1.6 - O - UNUSED
```

```
93        * P1.7 - O - UNUSED
94        * P2.0 - O - UNUSED
95        * P2.1 - O - Gate Driver MPPT MOSFET - TA1.1
96        * P2.2 - O - UNUSED
97        * P2.3 - O - UNUSED
98        * P2.4 - O - Gate Drive V-OUT MOSFET - TA1.2
99        * P2.5 - O - UNUSED
100       * P2.6 - O - UNUSED
101       * P2.7 - O - UNUSED
102       */
103
104     /*
105      * Configure GPIO
106      */
107     // Set P2.1 and P2.4 to output direction
108     P2DIR = (BIT1 | BIT4);
109     // Set P2.1 and P2.4 to primary function
110     P2SEL = (BIT1 | BIT4);
111     /*
112      * Configure unused pins
113      */
114     // Set unused P1 pins to output
115     P1DIR = (BIT3 | BIT6 | BIT7);
116     // Enable pull-up resistor for P1.1
117     P1REN = (BIT1 | BIT2);
118     // Setting BIT1 uses the Pull-Up resistor instead of the pull down resistor
119     P1OUT = (BIT1 | BIT2 | BIT3 | BIT7);
120     // Set unused P2 pins to output
121     P2DIR |= (BIT0 | BIT2 | BIT3 | BIT5 | BIT6 | BIT7);
122     // Set all P3 pins to output
123     P3DIR = 0xF;
124     P3OUT = 0x0;
125     P3REN = 0xF;
126
127     /*
128      * Configure ADC - Run on MCLK - 16MHz/4 = 4MHzx64 = 16uS
129      */
130     // Use Vcc and Vss, S&H Time: 64*ADC10CLKs, turn on ADC, enable interrupt
131     ADC10CTL0 = (SREF_0 | ADC10SHT_3 | ADC10ON | ADC10SR | ADC10IE);
132     // Use MCLK/4
133     ADC10CTL1 = (ADC10SSEL_2 | ADC10DIV_3);
134     // Using A0, A4, and A5
135     ADC10AE0 = (BIT0 | BIT4 | BIT5);
136     // Not using the DTC
137
138     /*
139      * Configure Timer1 - SMCLK
```

```
140         * PWM at 50KHz with 16MHz Clock
141         * Count to 320-1
142         */
143        // Use SMCLK, /1 divider, Up mode
144        TA1CTL = (TASSEL_2 | ID_0 | MC_1);
145        // Compare 1 should reset/set TA1.1
146        TA1CCTL1 = OUTMOD_7;
147        // Compare 2 should reset/set TA1.2
148        TA1CCTL2 = OUTMOD_7;
149        // PWM frequency should be 50KHz
150        TA1CCR0 = (MAX_DUTY_CYCLE - 1);
151
152        /*
153         * Configure Timer0 - ACLK
154         */
155        // Use ACLK, /1 divider, Up mode
156        TA0CTL = (TASSEL_1 | ID_0 | MC_1 | TAIE);
157        // 12KHz clock, into 3 gives 4KHz
158        TA0CCR0 = (LOW_FQ_CLOCK);
159
160        //Global Interrupt Enable
161        _BIS_SR(GIE);
162
163        // Iterator variable because this isn't c99 apparently
164        unsigned int i;
165
166        // Code Body
167        while (1) {
168            switch (mppt_state) {
169                case MPPT_WAIT:
170                    if (DCTL & MPPT_CONTROL) {
171                        // Mark that we've moved on
172                        DCTL &= ~MPPT_CONTROL;
173                        // If we have fresh data, move on to limit run speed
174                        mppt_state = MPPT_LIMIT_SPEED;
175                    }
176                    break;
177                case MPPT_LIMIT_SPEED:
178                    // Desired Frequency / (12KHz / TA0CCR0 / 8)
179                    if (slow_down >= MPPT_LIMITER) {
180                        slow_down = 0;
181                        // Call button handler
182                        button_handler();
183                        // Check for low voltage
184                        mppt_state = MPPT_LOW_VOLT_HANDLER;
185                    } else {
186                        slow_down++;
```

```
187                     mppt_state = MPPT_WAIT;
188                 }
189                 break;
190             case MPPT_LOW_VOLT_HANDLER:
191                 if (DCTL & INPUT_VOLTAGE_PRESENT) {
192                     // Calculate new data before Low Volt check
193                     mppt_state = MPPT_AVERAGE;
194                 } else {
195                     // Handle Low Volt flag
196                     mppt_state = MPPT_LOW_VOLT_DETECTED;
197                 }
198                 break;
199             case MPPT_LOW_VOLT_DETECTION:
200                 // Check if voltage is low before proceeding
201                 if (v_mppt < (V_MARGIN)) {
202                     if (zero_samples >= 20) {
203                         zero_samples = 0;
204                         // Not detecting a voltage, lets wait for the next run
205                         DCTL &= ~INPUT_VOLTAGE_PRESENT;
206                         mppt_state = MPPT_WAIT;
207                         break;
208                     } else {
209                         zero_samples++;
210                     }
211                 }
212                 // Voltage is still good, run algorithm
213                 mppt_state = MPPT_ALGORITHM;
214                 break;
215             case MPPT_LOW_VOLT_DETECTED:
216                 zero_samples++;
217                 if (zero_samples >= 20) {
218                     DCTL |= INPUT_VOLTAGE_PRESENT;
219                     // Zero integrals to avoid unnecessarily integrated error
220                     v_out_integral = 0;
221                     zero_samples = 0;
222                     // Call MPPT algorithm check for output voltage
223                     mppt_state = MPPT_AVERAGE;
224                     break;
225                 }
226                 mppt_state = MPPT_WAIT;
227                 break;
228             case MPPT_AVERAGE:
229                 // Get average current and voltage
230                 i_mppt = v_mppt = 0;
231                 for (i = AVERAGE_LENGTH; i > 0; i--) {
232                     // Calculate average I-MPPT
233                     i_mppt += i_mppt_samples[i - 1];
```

```
234                     // Calculate average V-MPPT
235                     v_mppt += v_mppt_samples[i - 1];
236                 }
237                 i_mppt = i_mppt >> AVERAGE_LENGTH_BIT;
238                 v_mppt = v_mppt >> AVERAGE_LENGTH_BIT;
239                 // Now that we have samples, check if voltage present
240                 mppt_state = MPPT_LOW_VOLT_DETECTION;
241                 break;
242             case MPPT_ALGORITHM:
243                 // Run MPPT algorithm (Set duty cycle - TA1CCR1)
244                 call_algorithm();
245                 mppt_state = MPPT_WAIT;
246                 break;
247             default:
248                 break;
249         }
250
251         switch (vout_state) {
252             case VOUT_WAIT:
253                 // Sit until we've collected new data
254                 if (DCTL & VOUT_CONTROL) {
255                     vout_state = VOUT_LOW_VOLT_HANDLER;
256                 }
257                 break;
258             case VOUT_LOW_VOLT_HANDLER:
259                 if (DCTL & INPUT_VOLTAGE_PRESENT) {
260                     vout_state = VOUT_AVERAGE;
261                 } else {
262                     vout_state = VOUT_DISABLE;
263                 }
264                 break;
265             case VOUT_AVERAGE:
266                 v_out = 0;
267                 for (i = AVERAGE_LENGTH; i > 0; i--) {
268                     v_out += v_out_samples[i - 1];
269                 }
270                 v_out = v_out >> AVERAGE_LENGTH_BIT;
271                 // Call control algorithm after computing average of samples
272                 vout_state = VOUT_ALGORITHM;
273             case VOUT_ALGORITHM:
274                 // Run Vout Control algorithm
275                 TA1CCR2 += adjust_output_duty_cycle(v_out, V_SETPOINT,
276                         &v_out_sat, &v_out_integral, v_out_i, Divisor);
277                 if (TA1CCR2 >= MAX_DUTY_CYCLE) {
278                     TA1CCR2 = MAX_DUTY_CYCLE;
279                 }
280                 break;
```

```
281             case VOUT_DISABLE:
282                 TA1CCR2 = 0;
283                 vout_state = VOUT_WAIT;
284                 break;
285         }
286     }
287 }
288
289 /** \brief Timer A0 ISR that starts ADC measurements
290  *
291  * Function starts ADC CH2 (V-OUT) measurement when TA0 rolls over.
292  * The period is controlled by TA0CCR0
293  */
294 #pragma vector=TIMER0_A1_VECTOR
295 __interrupt void timerA0_ISR(void) {
296     if (TA0IV == 0xA) {
297         // TACCR1
298         // Read        A4 / V-OUT
299         ADC10CTL0 &= ~ENC;
300         ADC10CTL1 &= 0xFFF;
301         ADC10CTL1 |= INCH_4;
302         // Start ADC conversion
303         ADC10CTL0 |= (ENC | ADC10SC);
304         periodic_timer_count++;
305     }
306 }
307
308 /** \brief ADC10 ISR stores measurement, and kicks-off next one if necessary.
309  *
310  * Function copies measurement value into the respective global variable.
311  * Kicks off V-MPPT measurement, then I-MPPT measurement. Also calls
312  * adjust_out_duty_cycle to control D-OUT.
313  */
314 #pragma vector=ADC10_VECTOR
315 __interrupt void ADC10_ISR(void) {
316     switch (ADC10CTL1 >> 12) {
317         // I-MPPT
318         case (0x0):
319             i_mppt_samples[sample] = ADC10MEM;
320             // Only update Duty cycle at 500Hz
321             // Increment sample count, roll over at 3
322             sample++;
323             if (sample == AVERAGE_LENGTH) {
324                 sample = 0;
325                 DCTL |= MPPT_CONTROL;
326             }
327             break;
```

```
328         // V-MPPT
329         case (0x5):
330             v_mppt_samples[sample] = ADC10MEM;
331             // Read        A0 / I-MPPT
332             ADC10CTL0 &= ~ENC;
333             ADC10CTL1 &= 0xFFF;
334             ADC10CTL1 |= INCH_0;
335             // Start ADC conversion
336             ADC10CTL0 |= (ENC | ADC10SC);
337             break;
338         // V-OUT
339         case (0x4):
340             v_out_samples[sample] = ADC10MEM;
341             // Read        A5 / V-MPPT
342             ADC10CTL0 &= ~ENC;
343             ADC10CTL1 &= 0xFFF;
344             ADC10CTL1 |= INCH_5;
345             // Start ADC conversion
346             ADC10CTL0 |= (ENC | ADC10SC);
347             // Enable D-OUT algorithm at 500Hz
348             if (sample == 0) {
349                 DCTL |= VOUT_CONTROL;
350             }
351             break;
352     }
353 }
354
355 /** \brief Adjusts the Duty Cycle of output buck converter
356  *
357  * Function adjusts the Duty Cycle of the output buck converter based on the
358  * measured voltage and the set-point. Uses a proportional algorithm to adjust
359  * duty cycle.
360  */
361 int adjust_output_duty_cycle(int input, int setpoint, signed char *sat,
362         long *x_integral, int Ki2, int n) {
363     int e = setpoint - input;
364     int x;
365
366     /* If there isn't saturation, or there is,
367      * but saturation is the opposite direction from the error
368      */
369     if (! ((*sat < 0 && e < 0) || (*sat > 0 && e > 0))) {
370         *x_integral = *x_integral + (long) Ki2 * e;
371         // Keep integral within range
372         if (*x_integral > LONG_MAX) {
373             *x_integral = LONG_MAX;
374             *sat = 1;
```

```
375         } else if (*x_integral < LONG_MIN) {
376             *x_integral = LONG_MIN;
377             *sat = -1;
378         } else {
379             *sat = 0;
380         }
381         x = (e >> n) + (int) (*x_integral >> 16);
382     }
383     return x;
384 }
385 
386 /** \brief Changes the current algorithm
387  *
388  * This function cycles through the implemented algorithms.
389  */
390 void change_algorithm() {
391     switch (algorithm) {
392         case MPPT_SWEEP:
393             algorithm = MPPT_PERTURBOBSERVE;
394             break;
395         case MPPT_PERTURBOBSERVE:
396             algorithm = MPPT_SWEEP;
397             break;
398         case MPPT_BETA:
399             break;
400         default:
401             break;
402     }
403 }
404 
405 /** \brief Calls algorithm reset function
406  *
407  * This function calls the chosen algorithm's reset function
408  */
409 void reset_algorithm() {
410     switch (algorithm) {
411         case MPPT_SWEEP:
412             sweep_reset(&DCTL);
413             break;
414         case MPPT_PERTURBOBSERVE:
415             perturb_and_observe_reset();
416             break;
417         case MPPT_BETA:
418             //beta_reset(&DCTL);
419             break;
420         default:
421             break;
```

```
422     }
423 }
424
425 /** \brief Calls algorithm function
426  *
427  * This function calls the chosen algorithm's main function
428  */
429 void call_algorithm() {
430     switch (algorithm) {
431         case MPPT_SWEEP:
432             TA1CCR1 = sweep(&DCTL);
433             break;
434         case MPPT_PERTURBOBSERVE:
435             TA1CCR1 = perturb_and_observe(&DCTL);
436             break;
437         case MPPT_BETA:
438 //          TA1CCR1 = beta(&DCTL);
439             break;
440         default:
441             break;
442     }
443 }
444
445 /** \brief Handles debouncing buttons
446  *
447  * This function handles debouncing buttons and calling the appropriate
448  * button handler
449  */
450 void button_handler() {
451     //Button handling
452     if ((P1IN & BIT1) == 0) {
453         BUTTONS |= RESET_BUTTON_PRESSED;
454     } else if (BUTTONS & RESET_BUTTON_PRESSED) {
455         // Button is no longer pressed
456         BUTTONS &= ~RESET_BUTTON_PRESSED;
457         reset_algorithm();
458     }
459     if ((P1IN & BIT2) == 0) {
460         BUTTONS |= ALGORITHM_BUTTON_PRESSED;
461     } else if (BUTTONS & ALGORITHM_BUTTON_PRESSED) {
462         // Button is no longer pressed
463         BUTTONS &= ~ALGORITHM_BUTTON_PRESSED;
464         change_algorithm();
465     }
466 }
```

```
/*******************************************************************************
 *                    Maximum Power Point Tracker (MPPT)
 *
 * This file implements the sweep method of finding and staying at the MPPT.
 *
 * This project is in fulfillment of the Major Qualifying Project at
 * Worcester Polytechnic Institute.
 *
 * Authored by:
 *         Jonathan Adams
 *         Ben Beauregard
 *         Andrew Flynn
 *
 ******************************************************************************/

// Includes
#include "main.h"
#include "sweep.h"
#include "msp430.h"


int sweep(volatile char *DCTL) {

    if (periodic_timer_count >= SWEEP_INTERVAL) {
        // Reset timer and sweep again
        periodic_timer_count = 0;
        sweep_reset(DCTL);
    }
    // Only need to do something if sweep is not complete
    if (sweep_complete == 0 ) {
        power = (long) i_mppt * v_mppt;
        // New power > old power, save duty cycle
        if (power > max_power) {
            max_power_duty_cycle = mppt_duty_cycle;
            max_power = power;
        }
        // We've reached 100% duty cycle, mark as complete,
        // Set duty cycle to maximum power point
        if (mppt_duty_cycle == 320) {
            sweep_complete = 1;
            mppt_duty_cycle = max_power_duty_cycle;
        } else {
            // Haven't completed sweep yet, increment duty cycle
            mppt_duty_cycle += SWEEPINC;
            /* TESTING CODE */
            if (mppt_duty_cycle % 16 == 0) {
                // Turn on LED at P1.6 every 5% duty cycle
```

```
48                  P1OUT |= (BIT6);
49              } else {
50                  // Turn off LED at P1.6 on next cycle
51                  P1OUT &= (~BIT6);
52              }
53              /* TESTING CODE */
54          }
55      }
56      return mppt_duty_cycle;
57  }
58
59  /** \brief Sweep reset called when button is pushed
60   *
61   * Function handles restarting sweep, including resetting variables
62   */
63  void sweep_reset(volatile char *DCTL) {
64      max_power = 0;
65      mppt_duty_cycle=80;
66      sweep_complete = 0;
67  }
```

```
/*******************************************************************************
 *                    Maximum Power Point Tracker (MPPT)
 *
 * This file implements the perturb and observe method of finding and staying
 * at the MPPT.
 *
 * This project is in fulfillment of the Major Qualifying Project at
 * Worcester Polytechnic Institute.
 *
 * Authored by:
 *         Jonathan Adams
 *         Ben Beauregard
 *         Andrew Flynn
 *
 ******************************************************************************/
#include "perturbobserve.h"
#include "main.h"
#include "msp430.h"

int perturb_and_observe(volatile char *DCTL) {
    power = (long)i_mppt * v_mppt;
    // Only adjust duty cycle if the difference was "significant"
    if ( (power >= (prev_power + PERTURBDEADZONE) ) ||
            (power <= (prev_power - PERTURBDEADZONE) ) ) {
        // Power decreased, lets change direction
        if (power < prev_power) {
            // Swap the direction
            direction = !direction;
        }
        if (mppt_duty_cycle == 0) {
            // If we're at zero duty cycle we need to increase the duty cycle
            direction = 1;
        } else if (mppt_duty_cycle == MAX_DUTY_CYCLE) {
            // If we're at 100% duty cycle we need to decrease the duty cycle
            direction = 0;
        }
        // Adjust duty cycle depending on direction flag
        switch (direction) {
            case 1:
                mppt_duty_cycle += PERTURBINC;
                // Turn on LED at P1.6 if we're increasing
                P1OUT |= BIT6;
                break;
            default:
                mppt_duty_cycle -= PERTURBINC;
                // Turn off LED at P1.6 if we're decreasing
                P1OUT &= ~BIT6;
```

```
48                  break;
49          }
50      }
51      prev_power = power;
52      return mppt_duty_cycle;
53  }
54
55  void perturb_and_observe_reset(void) {
56      // Reset direction to up
57      direction = 1;
58      // Reset previous power to 0 to ensure we move upwards
59      prev_power = 0;
60      // Start at 25% duty cycle
61      mppt_duty_cycle=80;
62  }
```

```c
/*******************************************************************************
 *                    Maximum Power Point Tracker (MPPT)
 *
 * This file implements the beta method of finding and staying at the MPPT.
 *
 * This project is in fulfillment of the Major Qualifying Project at
 * Worcester Polytechnic Institute.
 *
 * Authored by:
 *         Jonathan Adams
 *         Ben Beauregard
 *         Andrew Flynn
 *
 *******************************************************************************/

#include "math.h"
#include "main.h"
#include "beta.h"
#include "perturbobserve.h"

int beta(volatile char *DCTL) {
    // Store previous value
    beta_calculated_prev = beta_calculated;
    // Calculate current Beta value
    beta_calculated = logf(i_mppt/v_mppt) - CCONST * v_mppt;
    if ( (beta_calculated > BMAX) || (beta_calculated < BMIN) ){
        // If beta value far away from optimal value, calculate new duty cycle
        if ( (beta_calculated_prev < BMAX) && (beta_calculated_prev > BMIN) ){
            // Step size is adjusted from conditions at maximum power point
            beta_step_const = MAX_DUTY_CYCLE / (beta_calculated - BETADESIRED);
        }
        // Calculate new duty cycle from previous duty cycle and error
        mppt_duty_cycle += ((BETADESIRED - beta_calculated) * beta_step_const);
        // Limit duty cycle to correct range
        if (mppt_duty_cycle > MAX_DUTY_CYCLE) {
            mppt_duty_cycle = MAX_DUTY_CYCLE;
        } else if (mppt_duty_cycle < 0) {
            mppt_duty_cycle = 0;
        }
    } else {
        // Otherwise use another algorithm to reach the top
        perturb_and_observe(DCTL);
        beta_step_const = 0;
    }
    return mppt_duty_cycle;
}
```

G MATLAB Source Code

```
%% MATLAB script to prove functionality of MPPT algorithm
% Authors: Andrew Flynn, Ben Beauregard, Johnathan Adams
%
% This script attempts to imulate the functionality of the maximum power
% point tracking algorithm built in C for use with the projects hardware.
% This is meant as a proof that the concept is sound and the algorithm will
% track theoretical and actual maximum power point data, not as an actual
% implementation to run on the hardware. As such, certain things are
% abstracted, such as duty cycle setting and tracking, ISRs, and other
% MCU-specific functionality.

%% Flynn's Code to load Experimental Data
% We're just calling another script here.  It reads a data file in a
% terribly... unintuitive way, but it does work, and if it ain't broke...
clear;
fid = fopen('Full_Data.txt', 'r') ; % Open source file.
if fid == -1
    disp('ERROR 404: File Not Found') ;
else
    fgetl(fid) ;                                % Read/discard line.
    fgetl(fid) ;                                % Read/discard line.
    buffer = fread(fid, Inf) ;                  % Read rest of the file.
    fclose(fid);
    fid = fopen('_temp.txt', 'w')  ;   % Open destination file.
    fwrite(fid, buffer) ;                       % Save to file.
    fclose(fid) ;
    A = tdfread('_temp.txt') ;
    delete('_temp.txt') ;
end

%% Data-Error Check
% If we have more voltage samples than current samples something is wrong
% and should probably be fixed before the simulation is continued.
if length(A.Voltage_0x5BV0x5D) ~= length(A.x0x23_Current_0x5BA0x5D)
    error('Sample Size Mismatch. What are you doing?')
else
   IV = [A.Voltage_0x5BV0x5D.'; A.x0x23_Current_0x5BA0x5D.'];
end

%% Declaration of Variables
% These Variables are hereby independant from the monarch of Great Britan.
sim_time = 0; % tracks current elapsed time in simulation, us
Tsample = 1 / 500; % 500Hz MPPT sample rate, we want the period though
num_samp = length(IV); % Number of samples in performance data
delta_dc = 100 * (2 / 320); % Duty cycle increment value
```

```
46  duty_cycle = delta_dc; % Current duty cycle in simulation (range 0 - 100)
47  power = 0; % Holds current power point
48  mpp_dc = 0; % duty cycle at maximum power point
49  mpp = 0; % Power level at maximum power point
50  samp_idx = 1; % Array Index of current voltage/current sample
51  sim_data = []; % Array to store simulation results.
52  finished = 0; % Flag set when sweep is finished to prevent further sweeping
53
54  %% Data Resolution Error Check
55  % Confirm that we have enough experimental data to support the duty cycle
56  % increment value given to the simulation.  If not, abort, because
57  % simulation won't give accurate results.
58  if length(A.Voltage_0x5BV0x5D) < 100 / delta_dc
59      error(['Not enough simulation data points to match given duty '...
60          'cycle change rate.']);
61  end
62
63  %% Define Simulation Parameters
64  % These define the nature of the time-domain loop.
65  % This is not robust at all.  Simulation time must be evenly divisible by
66  % Tsample or simulation will derp.
67  disp(['Recommended Simulation Time (1 full sweep): ',...
68      num2str(Tsample * 160 * 1e3),'ms.']);
69  sim_len = input('Please enter the time length of the simulation, in ms: ');
70  while sim_len < Tsample * 10e3
71      tmpstr = ['Sim. Length too small.  Minimum length is ',...
72          num2str(Tsample * 10e3),'. Please try again: '];
73      sim_len = input(tmpstr);
74  end
75  sim_res = input('Please enter the time step for the simulation, in us: ');
76  while sim_res > Tsample * 1e6 / 2 || sim_res < .001
77      tmpstr = ['Resolution invalid.  Maximum resolution is ',...
78          num2str(Tsample * 1e6 / 2),...
79          'us, minimum is 0.001us. Please try again: '];
80      sim_res = input(tmpstr);
81  end
82
83  %% Time-Domain Simulation Loop
84  % While the simulation time is less than the requested time, continue
85  % simulating.
86  while sim_time < sim_len * 1e-3
87      % Change Duty Cycle at Given Interval
88      % mod(x,y) doesn't seem to give actual zero answers... derp...
89      if mod(sim_time,Tsample) < 1e-12 && ~ finished
90          % Increment Duty Cycle
91          if duty_cycle >= 100 - delta_dc
92              duty_cycle = mpp_dc;
```

```
93              finished = 1; % Mark end of sweep
94          else
95              duty_cycle = duty_cycle + delta_dc;
96          end
97          % Check if MPP
98          % This indexing code is a bit wat
99          samp_idx = round(length(A.Voltage_0x5BV0x5D) * duty_cycle / 100);
100         power = A.Voltage_0x5BV0x5D(samp_idx) *...
101             A.x0x23_Current_0x5BA0x5D(samp_idx);
102         if power > mpp
103             mpp = power;
104             mpp_dc = duty_cycle;
105         end
106     end
107     % Data Logging
108     sim_data = [sim_data;[sim_time,duty_cycle,power,...
109         A.Voltage_0x5BV0x5D(samp_idx),...
110         A.x0x23_Current_0x5BA0x5D(samp_idx)]]; % Rewrite this later
111     % Increment simulaiton time and restart loop
112     sim_time = sim_time + sim_res * 1e-6;
113 end
114
115 %% Graph the Simulation Results
116 [Ax,H1,H2] = plotyy(sim_data(:,1),sim_data(:,2),sim_data(:,1),...
117     sim_data(:,3));
118 title('Power and Duty Cycle versus Time');
119 xlabel('Time, [s]');
120 ylabel(Ax(1),'Duty Cycle, [%]');
121 ylabel(Ax(2),'Power, [W]');
```

H Datasheet Excerpts

H.1 MIC5021 High-Side MOSFET Driver

MICROCHIP

MIC5021

High-Speed, High-Side MOSFET Driver with Charge Pump and Overcurrent Limit

Features

- 12V to 36V Operation
- 550 ns Rise/Fall Time Driving 2000 pF
- TTL-Compatible Input with Internal Pull-Down Resistor
- Overcurrent Limit
- Gate-to-Source Protection
- Internal Charge Pump
- 100 kHz Operation Guaranteed Over Full Temperature and Operating Voltage Range
- Compatible with Current-Sensing MOSFETs
- Current-Source Drive Reduces EMI

Applications

- Lamp Control
- Heater Control
- Motor Control
- Solenoid Switching
- Switch-Mode Power Supplies
- Circuit Breaker

General Description

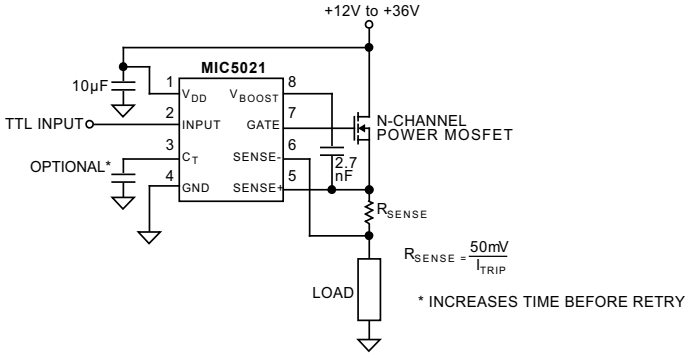
The MIC5021 high-side MOSFET driver is designed to operate at frequencies up to 100 kHz (5 kHz PWM for 2% to 100% duty cycle) and is an ideal choice for high speed applications such as motor control, SMPS (switch mode power supplies), and applications using IGBTs. The MIC5021 can also operate as a circuit breaker with or without automatic retry.

A rising or falling edge on the input results in a current source pulse or sink pulse on the gate output. This output current pulse can turn on a 2000 pF MOSFET in approximately 550 ns. The MIC5021 then supplies a limited current (<2 mA), if necessary, to maintain the output state.

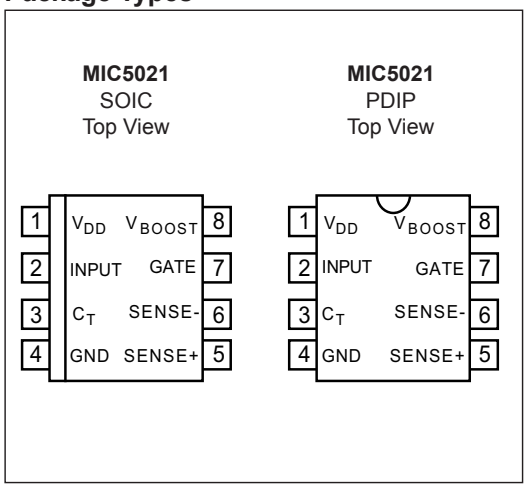
An overcurrent comparator with a trip voltage of 50 mV makes the MIC5021 ideal for use with a current-sensing MOSFET. An external low value resistor may be used instead of a sensing MOSFET for more precise overcurrent control. An optional external capacitor placed from the C_T pin to ground may be used to control the current shutdown duty cycle (dead time) from 20% to <1%. A duty cycle from 20% to about 75% is possible with an optional pull-up resistor from C_T to V_{DD}. Additional parts of the MIC502x family include the MIC5020 low-side driver and the MIC5022 half-bridge driver with a cross-conduction interlock. The MIC5021 is available in 8-pin SOIC and plastic DIP packages.

Typical Application Circuit

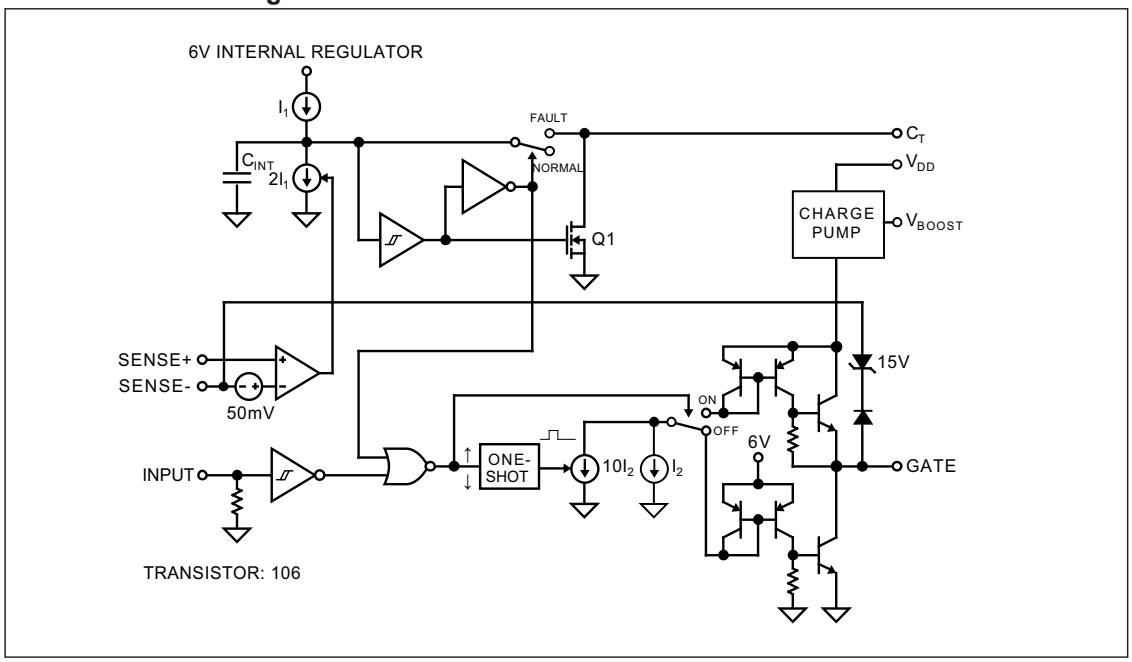
MIC5021
PDIP & SOIC
High-Side Driver with Overcurrent Trip and Retry



Package Types



Functional Block Diagram



1.0 ELECTRICAL CHARACTERISTICS

Absolute Maximum Ratings †

Supply Voltage, V_{DD} ..... +40V
Input Voltage, V_{IN} ..... –0.5V to +15V
Sense Differential Voltage ..... ±6.5V
SENSE+ or SENSE– to GND ..... –0.5V to +36V
Timer Voltage ..... +5.5V
V_{BOOST} Capacitor ..... 0.01 µF

Operating Ratings

Supply Voltage, V_{DD} ..... +12V to +36V

† Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operational sections of this specification is not intended. Exposure to maximum rating conditions for extended periods may affect device reliability.

DC CHARACTERISTICS

Electrical Characteristics: Unless otherwise indicated, T_A = +25°C, GND = 0V, V_{DD} = 12V, C_T = OPEN, Gate C_L = 1500 pF (IRF540 MOSFET).

Parameters	Sym.	Min.	Typ.	Max.	Units	Conditions
DC Supply Current	—	—	1.8	4	mA	V_{DD} = 12V, Input = 0V
	—	—	2.5	6		V_{DD} = 36V, Input = 0V
	—	—	1.7	4		V_{DD} = 12V, Input = 5V
	—	—	2.5	6		V_{DD} = 36V, Input = 5V
Input Threshold	—	0.8	1.4	2.0	V	—
Input Hysteresis	—	—	0.1	—	V	—
Input Pull-Down Current	—	10	20	40	µA	Input = 5V
Current-Limit Threshold	—	30	50	70	mV	Note 1
Gate On Voltage	—	16	18	21	V	V_{DD} = 12V (Note 2)
	—	46	50	52		V_{DD} = 36V (Note 2)
Gate On-Time (Fixed)	$t_{G(ON)}$	2	6	10	µs	Sense Differential > 70 mV (Note 8)
Gate Off-Time (Adjustable)	$t_{G(OFF)}$	10	20	50	µs	Sense Differential > 70 mV, C_T = 0 pF (Note 8)
Gate Turn-On Delay	t_{DLH}	—	500	1000	ns	Note 3
Gate Rise Time	t_R	—	400	500	ns	Note 4
Gate Turn-Off Delay	t_{DLH}	—	800	1500	ns	Note 5

Note 1: When using sense MOSFETs, it is recommended that R_{SENSE} < 50Ω. Higher values may affect the sense MOSFET's current transfer ratio.

2: DC measurement.

3: Input switched from 0.8V (TTL low) to 2.0V (TTL high), time for gate transition from 0V to 2V.

4: Input switched from 0.8V (TTL low) to 2.0V (TTL high), time for gate transition from 2V to 17V.

5: Input switched from 2.0V (TTL high) to 0.8V (TTL low), time for gate transition from 20V (gate on voltage) to 17V.

6: Input switched from 2.0V (TTL high) to 0.8V (TTL low), time for gate transition from 17V to 2V.

7: Frequency where gate on voltage reduces to 17V with 50% input duty cycle.

8: Gate on time $t_{G(ON)}$ and $t_{G(OFF)}$ are not 100% production tested.

DC CHARACTERISTICS (CONTINUED)

Electrical Characteristics: Unless otherwise indicated, T_A = +25°C, GND = 0V, V_{DD} = 12V, C_T = OPEN, Gate C_L = 1500 pF (IRF540 MOSFET).

Parameters	Sym.	Min.	Typ.	Max.	Units	Conditions
Gate Fall Time	t_F	—	400	500	ns	Note 6
Max. Operating Frequency	f_{MAX}	100	150	—	kHz	Note 7

Note 1: When using sense MOSFETs, it is recommended that R_{SENSE} < 50Ω. Higher values may affect the sense MOSFET's current transfer ratio.

2: DC measurement.

3: Input switched from 0.8V (TTL low) to 2.0V (TTL high), time for gate transition from 0V to 2V.

4: Input switched from 0.8V (TTL low) to 2.0V (TTL high), time for gate transition from 2V to 17V.

5: Input switched from 2.0V (TTL high) to 0.8V (TTL low), time for gate transition from 20V (gate on voltage) to 17V.

6: Input switched from 2.0V (TTL high) to 0.8V (TTL low), time for gate transition from 17V to 2V.

7: Frequency where gate on voltage reduces to 17V with 50% input duty cycle.

8: Gate on time $t_{G(ON)}$ and $t_{G(OFF)}$ are not 100% production tested.

6.0 APPLICATION INFORMATION

The MIC5021 MOSFET driver is intended for high-side switching applications where overcurrent limiting and high speed are required. The MIC5021 can control MOSFETs that switch voltages up to 36V.

6.1 High-Side Switch Circuit Advantages

High-side switching allows more of the load related components and wiring to remain near ground potential when compared to low-side switching. This reduces the chances of short-to-ground accidents or failures.

6.2 Speed Advantage

The MIC5021 is about two orders of magnitude faster than the low cost MIC5014 making it suitable for high-frequency high-efficiency circuit operation in PWM (pulse width modulation) designs used for motor control, SMPS (switch-mode power supply) and heating element control.

Switched loads (on/off) benefit from the MIC5021's fast switching times by allowing use of MOSFETs with smaller safe operating areas. Larger MOSFETs are often required when using slower drivers.

6.3 Supply Voltage

The MIC5021's supply input (V_{DD}) is rated up to 36V. The supply voltage must be equal to or greater than the voltage applied to the drain of the external N-channel MOSFET.

A 16V minimum supply is recommended to produce continuous on-state, gate drive voltage for standard MOSFETs (10V nominal gate enhancement).

When the driver is powered from a 12V to 16V supply, a logic-level MOSFET is recommended (5V nominal gate enhancement).

PWM operation may produce satisfactory gate enhancement at lower supply voltages. This occurs when fast switching repetition makes the boost capacitor a more significant voltage supply than the internal charge pump.

6.4 Logic-Level MOSFET Precautions

Logic-level MOSFETs have lower maximum gate-to-source voltage ratings (typically ±10V) than standard MOSFETs (typically ±20V). When an external MOSFET is turned on, the doubling effect of the boost capacitor can cause the gate-to-source voltage to momentarily exceed 10V. Internal zener diodes clamp this voltage to 16V maximum which is too high for logic-level MOSFETs. To protect logic-level MOSFETs, connect a zener diode ($5V \le V_{ZENER} < 10V$) from gate to source.

6.5 Overcurrent Limiting

A 50 mV comparator is provided for current sensing. The low level trip point minimizes I_2R losses when a power resistor is used for current sensing.

The adjustable retry feature can be used to handle loads with high initial currents, such as lamps or heating elements, and can be adjusted from the C_T connection.

C_T to ground maintains gate drive shutdown following an overcurrent condition.

C_T open, or a capacitor to ground, causes automatic retry. The default duty cycle (C_T open) is approximately 20%. Refer to the Electrical Characteristics when selecting a capacitor for reduced duty cycle.

C_T through a pull-up resistor to V_{DD} increases the duty cycle. Increasing the duty cycle increases the power dissipation in the load and MOSFET under a fault condition. Circuits may become unstable at a duty cycle of about 75% or higher, depending on conditions. Caution: The MIC5021 may be damaged if the voltage applied to C_T exceeds the absolute maximum voltage rating.

6.6 Boost Capacitor Selection

The boost capacitor value will vary depending on the supply voltage range.

A 0.01 µF boost capacitor is recommended for best performance in the 12V to 20V range. (See Figure 6-1.) Larger capacitors may damage the MIC5021.

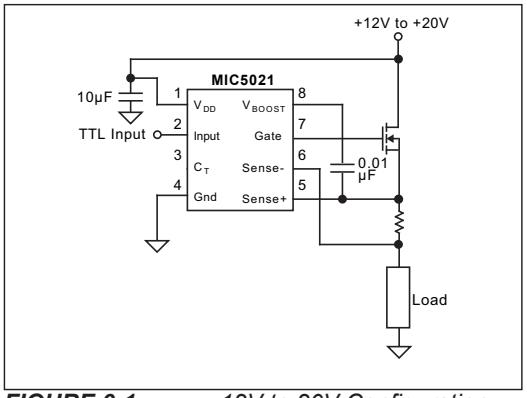


FIGURE 6-1: *12V to 20V Configuration.*

If the full 12V to 36V voltage range is required, the boost capacitor value must be reduced to 2.7 nF (Figure 6-2). The recommended configuration for the 20V to 36V range is to place the capacitor is placed between V_{DD} and V_{BOOST} as shown in Figure 6-3.

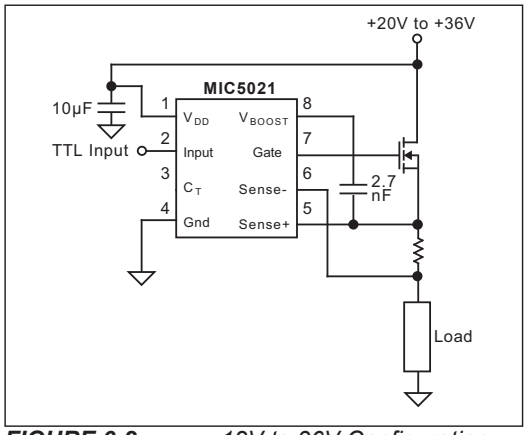


FIGURE 6-2: *12V to 36V Configuration.*

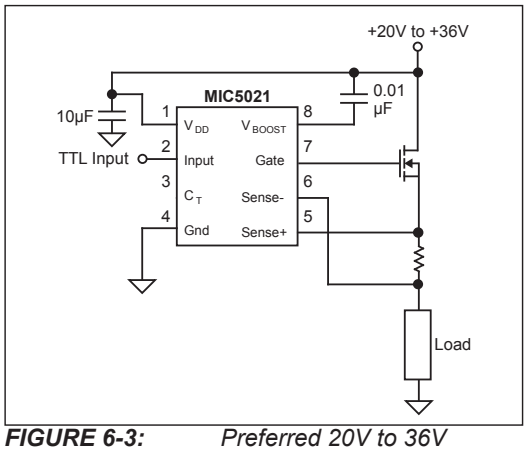


FIGURE 6-3: *Preferred 20V to 36V Configuration.*

Do not use both boost capacitors between V_{BOOST} and the MOSFET source and V_{BOOST} and V_{DD} at the same time.

6.7 Current-Sense Resistors

Lead length can be significant when using low value (<1Ω) resistors for current sensing. Errors caused by lead length can be avoided by using four-terminal current-sensing resistors. Four-terminal resistors are available from several manufacturers.

6.8 Circuits without Current Sensing

Current sensing may be omitted by connecting the SENSE+ and SENSE– pins to the source of the MOSFET or to the supply. Connecting the sense pins to the supply is preferred for inductive loads. Do not connect the sense pins to ground.

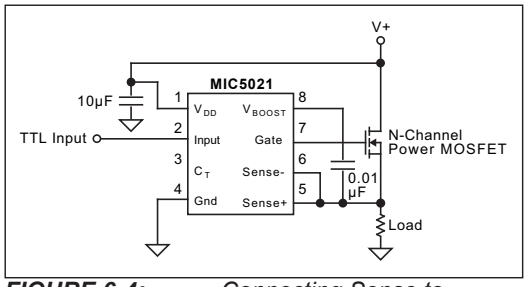


FIGURE 6-4: *Connecting Sense to Source.*

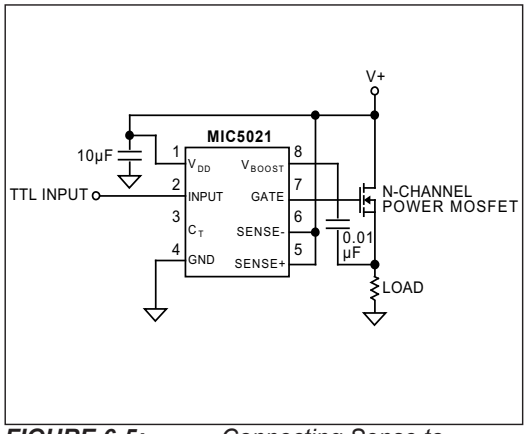


FIGURE 6-5: *Connecting Sense to Supply.*

6.9 Inductive Load Precautions

Circuits controlling inductive loads, such as solenoids (Figure 6-6) and motors, require precautions when controlled by the MIC5021. Wire wound resistors, which are sometimes used to simulate other loads, can also show significant inductive properties.

An inductive load releases stored energy when its current flow is interrupted (when the MOSFET is switched off). The voltage across the inductor reverses and the inductor attempts to force current flow. Since the circuit appears open (the MOSFET appears as a very high resistance) a very large negative voltage occurs across the inductor.

H.2 MSP430 Pinout Diagram

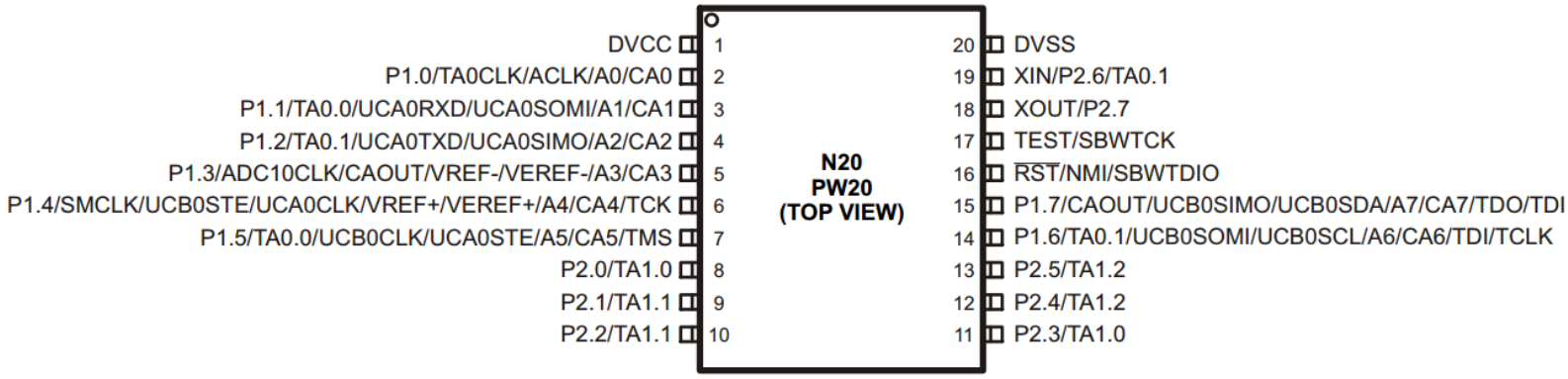


Figure H1: MSP430 Pinout Diagram from datasheet